ВВЕДЕНИЕ

В.1. Дисциплина программной инженерии

Разработка программ, как и многие другие виды деятельности человека, может производиться различными способами. Как ремесленник, использующий минимальный набор ручного инструмента, создаёт штучные изделия народного промысла, так и индивидуальный разработчик может воплощать свои идеи в программном продукте, используя лишь текстовый редактор и компилятор.

Но как современный мир был бы невозможен без массового промышленного производства, которое осуществляется на фабриках и заводах большими коллективами согласованно действующих людей с использованием разнообразных машин и механизмов, точно так же невозможна качественная реализация больших программных систем без применения специальных инструментов, следования установленным правилам организации рабочего процесса и взаимодействия участников проекта. В результате научных исследований, многочисленных проб и ошибок были выработаны приёмы разработки программ, позволяющие с высокой степенью вероятности гарантировать получение качественного программного продукта, отвечающего требованиям заказчиков, — сформировалась дисциплина программной инженерии.

Чтобы занять своё место в команде, программисту уже недостаточно знать методы алгоритмизации и язык программирования. Он должен владеть инструментарием, который используют участники команды для совместной работы над проектом. В число необходимых инструментов входит среда разработки (включающая редактор исходных текстов, компилятор, компоновщик и отладчик) и система контроля версий. Зачастую используются дополнительные инструменты — системы учёта заявок на доработку, автоматизированного тестирования, доставки и развёртывания программной системы на целевой платформе, где осуществляется её промышленная эксплуатация.

Эти актуальные требования к квалификации ИТ-специалистов нашли отражение в федеральных государственных образовательных стандартах (ФГОС) среднего профессионального (СПО) и высшего образования (ВО):

ФГОС СПО 09.02.03 — Программирование в компьютерных системах;

ФГОС ВО 09.03.02 — Информационные системы и технологии;

ФГОС ВО 09.03.04 — Программная инженерия.

В частности, в рамках междисциплинарного курса МДК 03.02 «Инструментальные средства разработки программного обеспечения» предъявляются требования к знанию учащимися структур и приёмов работы с инструментальными средствами, поддерживающими создание программного обеспечения, пониманию методов организации работы в коллективах разработчиков программного обеспечения. Студенты, осваивающие программу бакалавриата, должны научиться организовывать работу малых коллективов исполнителей, применять при решении задач профессиональной деятельности современные информационные технологии и программные средства, инсталлировать программное обес-

печение для информационных и автоматизированных систем. Программами подготовки ИТ-специалистов предусмотрены лабораторные работы, производственная и преддипломная практики для получения практических навыков использования инструментальных средств разработки программного обеспечения.

В.2. Инструменты системы Fossil

Эта книга посвящена бесплатному свободно распространяемому интегрированному программному продукту Fossil (http://fossil-scm.org), который реализует сразу пять инструментов, необходимых в коллективной работе над проектами: систему контроля версий (аналог Git), систему учёта заявок (аналог Jira), систему документирования (аналог Wiki), форум и статический веб-сервер. Некоторым пользователям не хватает терпения разобраться со всеми возможностями и особенностями применения системы Fossil, но тех, кто преодолел первые трудности, она покоряет своей простотой и изяществом архитектуры.

Будучи задуманной в качестве системы контроля версий для поддержки процесса разработки системы управления базами данных SQLite, система Fossil постепенно обрастала дополнительными подсистемами, которые способствовали улучшению процесса совместной работы, но которые базировались на одном и том же механизме хранения версионированной информации. В результате получился компактный набор инструментов, заключённый в одном-единственном исполняемом файле, который легко устанавливать и эксплуатировать. Благодаря этому на её основе могут быть развёрнуты учебные стенды для проведения практических и лабораторных работ по использованию систем контроля версий и систем управления заявками.

Система Fossil достаточно стабильна для промышленного применения. Однако, несмотря на то, что в переводе с английского слово fossil означает ископаемое и окаменелость, она продолжает динамично развиваться. Инструменты, входящие в состав этой системы, могут быть использованы студентами при выполнении курсовых и дипломных проектов, при прохождении производственных практик, а специалистами — в их профессиональной деятельности.

Возвращаясь к высказанной аналогии между ремесленниками и индивидуальными разработчиками, можно отметить, что в современном мире даже мастера народных промыслов предпочитают использовать усовершенствованные инструменты с электрическим приводом. Поэтому программистам, которые занимаются разработкой программных продуктов единолично, стоит присмотреться к инструментам Fossil и воспользоваться ими для улучшения организации своей деятельности.

Система Fossil представляет интерес не только как набор инструментов, но и как самостоятельный объект изучения. Она реализована командой высококвалифицированных разработчиков, которая занимается развитием проекта SQLite, на языке программирования Си, соответствующем стандарту C89. Система имеет ясную архитектуру, что позволяет студентам вносить изменения в исходные тексты, собирать из них исполняемые файлы и непосредственно наблюдать результат своих действий. В системе Fossil реализован режим работы

с базой данных репозитория на низком уровне, что позволяет использовать язык запросов SQL для изучения структуры репозитория. С помощью системы Fossil можно без погружения в сложное конфигурирование наглядно продемонстрировать работу веб-сервера.

Некоторые современные книги содержат многочисленные ссылки на дополнительные материалы, размещённые в сети Интернет. По мнению автора, книга — не самый лучший способ хранения интернет-закладок. Поэтому, в отличие от них, эта книга самодостаточна. В ней имеются все необходимые иллюстрации, полностью приводятся интерактивные диалоги между пользователем и системой Fossil в режиме командной строки. Книгу можно читать вообще без компьютера, и это не должно затруднить усвоение материала. К практическим упражнениям можно переходить после заочного знакомства с системой. Единственный необходимый веб-ресурс, который упоминается в книге, — это официальный веб-сайт системы Fossil, на котором можно получить актуальные версии дистрибутивов системы для различных операционных систем.

В связи с отсутствием устоявшейся русскоязычной терминологии в области систем контроля версий при написании книги большое внимание уделялось составлению тематического словаря (результат этой работы можно увидеть в глоссарии, который приведён в приложении).

Глава 1
ЭВОЛЮЦИЯ СИСТЕМ КОНТРОЛЯ ВЕРСИЙ

1.1. Важность хранения истории изменений

Начнём эту книгу с диалога, который вполне мог бы произойти.

— Представляешь, я потерял все свои наработки...

— Как это могло случиться?! Разве ты не делал резервные копии?

— Конечно, делал! Кроме файла на ноутбуке, у меня были его копии на флешке и в облачном хранилище.

— Что же тогда произошло?

— Понимаешь, в конце работы я решил подсчитать количество слов в документе. Выделил весь текст, увидел то, что мне было нужно, а потом, перед сохранением, наверное, нечаянно нажал какую-то клавишу — и выделенный текст пропал. Я этого не заметил и переписал эту копию и на флешку, и в хранилище. А сегодня утром смотрю — везде пустые файлы...

Действительно, одно только наличие резервных копий рабочих файлов не может гарантировать сохранность находящейся в них информации. Повредится эталон, не важно, в результате ошибочных действий или технического сбоя, — и битая копия затрёт результаты работы на резервных носителях.

Не удивительно, что люди стали искать способ решения этой потенциальной проблемы и придумали для этого множество способов. Одним из лучших решений для предотвращения утраты результатов труда является использование специального инструмента — системы контроля версий, которая не заменяет старые варианты файлов новыми, а хранит всю историю их изменений.

Если использовать не локальную, а распределённую сетевую систему контроля версий, то история работы будет храниться на нескольких компьютерах, что практически исключит риски, о которых шла речь в начале этого раздела. Эта книга посвящена одному из таких инструментов — распределённой системе контроля версий Fossil.

Кому может понадобиться система контроля версий? Если ответить на этот вопрос кратко, то всем, чья работа с компьютером заключается во внесении накапливающихся изменений в различные файлы до тех пор, пока не будет решена поставленная задача.

В эту категорию, прежде всего, попадают программисты и веб-разработчики. Их деятельность напрямую связана с написанием исходных текстов программ и веб-страниц. Они начинают с пустого текстового файла и постепенно заполняют его фразами на языке разработки, периодически оценивая промежуточные результаты. Если промежуточный результат соответствует ожидаемому, то проект продолжают развивать от достигнутого. В противном случае в текущие файлы вносят изменения, пока результат не станет соответствовать промежуточной цели.

Похожую методику используют писатели, работая над художественным произведением. Они последовательно развивают сюжетные линии своего рома-

на, но в процессе творческого поиска иногда заходят в тупик и, чтобы выбраться из него, возвращаются к одному из предыдущих вариантов черновика.

Юристы коммерческих и государственных структур, работая над проектами контрактов, распоряжений, приказов, постановлений, нормативных актов, законов обычно начинают с утверждённого шаблона. В ходе доработки документ проходит многочисленные согласования, в результате которых возникают поправки и корректировки. Этот процесс продолжается до получения окончательного варианта и его утверждения.

Системные администраторы и администраторы программных комплексов регулярно вносят изменения в конфигурационные файлы. Внедрив систему контроля версий, они прежде всего получат автоматически ведущийся журнал, с помощью которого не составит труда установить, какое изменение привело к возникновению проблем, а также когда, кем и почему оно было выполнено. Помимо электронного журнала у них появится возможность мгновенно восстановить одно из сохранённых состояний администрируемой системы.

Перечень специалистов, работа которых ведётся по аналогичным схемам, можно дополнить инженерами-конструкторами и проектировщиками, музыкантами и художниками, которые используют в своей профессиональной деятельности компьютер. Но убедить использовать новый инструмент профессионалов, которые много лет выполняют свою работу по сложившейся методике, непросто. Тем более если работа с ним требует обучения и привыкания, нарушает привычный ритм и последовательность операций.

Замечено, что улучшения рабочего процесса обычно дают положительный эффект по прошествии некоторого времени, а в ближайшей перспективе, наоборот, приводят к уменьшению производительности труда. Стоят ли усилия, которые необходимо приложить для внедрения нового инструмента, тех результатов, которые будут получены? Чтобы это оценить и лучше понять, какие проблемы и каким образом решают системы контроля версий, надо проследить историю их появления и развития. Но сначала рассмотрим типичные приёмы, которые применяются в работе с файлами без использования этих систем и которые базируются на здравом смысле и подручных средствах.

1.2. Здравый смысл и подручные средства

Как работает программист, который не использует систему контроля версий? Основной инструмент программиста — текстовый редактор, и работу над проектом он начинает с создания текстового файла, в который записывает фразы на языке программирования проекта. Перед этим, конечно, программист создал отдельный каталог для хранения файлов с исходными текстами программы. В дальнейшем этот каталог будем называть рабочим каталогом проекта.

Сначала работа продвигается бодро: каркас большинства программ выглядит одинаково, пальцы сами набирают хорошо известные конструкции. Через непродолжительное время вызывающая программа готова, прошла тестирование и полностью удовлетворяет программиста.

Опытный программист предпочитает фиксировать варианты исходных текстов, из которых собирается приемлемая на определённом этапе программа. Поэтому сейчас он сохраняет исходные тексты из рабочего каталога в другом каталоге, например с именем VER001. Понимая, что уже через час он не вспомнит, какой стадии проекта соответствует содержимое этого каталога, программист создаёт в рабочем каталоге проекта текстовый файл history.txt и вписывает в него первую строку:

```
11:18 09.03.2020 VER001 Каркас приложения.
```

По мере реализации различных функций программы на диске появляются каталоги VER002, VER003, ..., VER012, а текстовый файл history.txt дополняется строками:

```
18:03 16.02.2020 VER012 Исправлена ошибка выхода за границы масси-
ва при вычислении медианы.
…
09:20 10.02.2020 VER003 Реализованы действия, соответствующие под-
пунктам меню Настройки.
16:31 09.02.2020 VER002 Реализованы действия, соответствующие под-
пунктам меню Файл.
11:18 06.02.2020 VER001 Каркас приложения.
```

На каком-то этапе программист понимает, что при реализации очередной функции зашёл в тупик. И гораздо легче вернуться назад на несколько шагов к сохранённому варианту, чтобы продолжить развитие проекта в другом направлении, чем исправлять нагромождение созданного с того времени кода.

Казалось бы, нет ничего проще — переписать содержимое требуемого VER-каталога в рабочий каталог и работать как ни в чём не бывало. Но, с другой стороны, может быть, всё не так страшно, как кажется в данный момент? До попадания в тупиковую ситуацию было реализовано несколько интересных алгоритмов, которые могли бы быть полезны в дальнейшей работе, и не хотелось бы их потерять при откате к предыдущей версии.

Поэтому программист решает сохранить текущее, пусть и представляющееся сейчас бесперспективным, состояние проекта. Для этого он создаёт каталог BAD001, копирует туда файлы с исходными текстами из рабочего каталога и вносит запись в history.txt:

```
12:05 28.02.2020 BAD001 Тупик при реализации обхода дерева клиен-
тов - откат к VER026. Эффективно реализована группировка по воз-
растным группам.
```

Теперь кроме резервных копий успешно реализованных этапов проекта в каталогах VER### начинает выстраиваться линейка тупиковых вариантов в каталогах BAD###. И это не зря, потому что даже в показавшемся сначала бесперспективным варианте после испытания альтернативных решений может обнаружиться рациональное зерно.

Иногда способ реализации очередной функции программы не очевиден или у программиста имеется несколько идей, из которых на данном этапе развития проекта он не готов выбрать лучшую без предварительной апробации.

В этом случае программист создаёт каталог TEST и делает в нём предварительный набросок одного из возможных вариантов. Может быть, он создаст несколько каталогов: TEST1, TEST2, TEST3, — чтобы проверить свои идеи. Потом, выбрав из вариантов наилучший, перепишет соответствующие ему исходные тексты в рабочий каталог и продолжит разработку.

Так, шаг за шагом, в конце концов этот проект будет реализован. Оглядываясь назад, можно отметить основные организационные мероприятия, которые ему сопутствовали:

- создание мгновенных снимков рабочего каталога для использования в качестве резервных копий;
- аннотирование созданных мгновенных снимков краткими описаниями для того, чтобы в дальнейшем можно было найти требуемую точку сохранения;
- периодические возвраты к точкам сохранения;
- создание ответвлений в развитии проекта для тестирования различных вариантов решения задачи.

Все перечисленные организационные задачи программист решал средствами операционной системы, используя в качестве базы данных иерархическую файловую систему.

В рассмотренном примере программист разрабатывал программу единолично. На сегодняшний день такие случаи скорее исключение, чем правило. Даже в организациях, где создание программного продукта не является основным видом деятельности, над сопровождением эксплуатирующихся программ трудятся несколько человек, которые должны иметь одинаковые права на доступ к их исходным текстам и файлам конфигурации.

Коллективная работа над проектом вносит дополнительные трудности. Индивидуальность отдельных участников может проявиться самым непредсказуемым образом, различия в представлениях о лучшем способе решения задачи могут приводить к конфликтам в самых сплочённых группах.

Нельзя исключать ситуации, когда двум различным участникам проекта одновременно потребуется внести изменения в исходные тексты функций, находящихся в одном файле. В этом случае зафиксируется результат того специалиста, который сохранит свой вариант файла последним, а работа другого участника при этом будет утеряна.

Организационные меры в виде списка правил, которым должны следовать все участники, не всегда способны исправить ситуацию. Не удивительно, что начали проводиться исследования в поиске технических средств для улучшения процесса работы над проектом.

1.3. Локальные системы контроля версий

Первой системой контроля версий, получившей признание, стала система контроля исходных текстов SCCS (Source Code Control System). Она была разработана в Bell Laboratories и реализована на компьютерах IBM 370 и PDP 11, о чём сообщил в 1975 г. Марк Дж. Рочкинд (Marc J. Rochkind).

В то время уже реализовывались масштабные программные проекты, и их разработчики постоянно сталкивались с одними и теми же проблемами. Во-первых, имелась сложность восстановления истории модификаций исходных текстов программ. Очевидно, что если в какой-то момент разрабатываемая программа перестаёт работать так, как ожидается, то это происходит из-за каких-то изменений. Для быстрого решения подобных проблем важно знать, кем, когда, где и для чего были внесены эти изменения.

Во-вторых, лавинообразно возрастал объём дискового пространства, необходимый для хранения исходных текстов программ. Для каждого значительного изменения при выпуске новой версии продукта создавалась полная копия файлов рабочего каталога. Если для какого-то клиента требовалась специфическая функциональность, которую невозможно отразить с помощью имеющихся в программе настроек, то специально для него создавалась отдельная ветвь разработки. При этом различия между копиями исходных текстов были крайне малы по сравнению с их размерами, что позволяло судить о неэффективном расходовании памяти.

Система контроля исходных текстов SCCS решала обе эти проблемы. Контролируемые ею файлы были представлены последовательностью всех отдельных изменений, называемых дельтами, которые произошли с момента создания вплоть до актуального состояния. Чтобы получить файл в его текущем состоянии, надо пройти по всей последовательности, применяя имеющиеся в ней изменения. Если этот процесс остановить в промежуточной точке, то будет получена одна из предыдущих версий файла.

На этой последовательности особо отмечаются точки, в которых изменяется версия программы. К этим и только к этим точкам программист впоследствии может дописывать изменения, исправляющие обнаруженные ошибки. Изменения, выполненные в рамках «работы над ошибками», учитываются только при извлечении исходного текста интересующей версии, но исключаются из прямой цепочки изменений, ведущей к актуальному состоянию файла (рис. 1.1).

При использовании системы SCCS доступ к файлам с исходными текстами полностью контролируется этой системой. Для того, чтобы доработать необходимый файл, программист должен сначала запросить его у системы, затем внести необходимые изменения, после чего вернуть его в систему, сопроводив описанием проделанной работы. Система автоматически определяет различия между предыдущим и новым вариантами файла и сохраняет их в виде дельты, снабжая отметкой времени модификации, номером версии и именем автора изменений.

К началу 1980-х гг. сформировались современные теоретические основы систем контроля версий. История изменений объекта контроля представляется в виде направленного ацикличного графа (directed acyclic graph, DAG), вершины которого соответствуют промежуточным состояниям объекта (рис. 1.2).

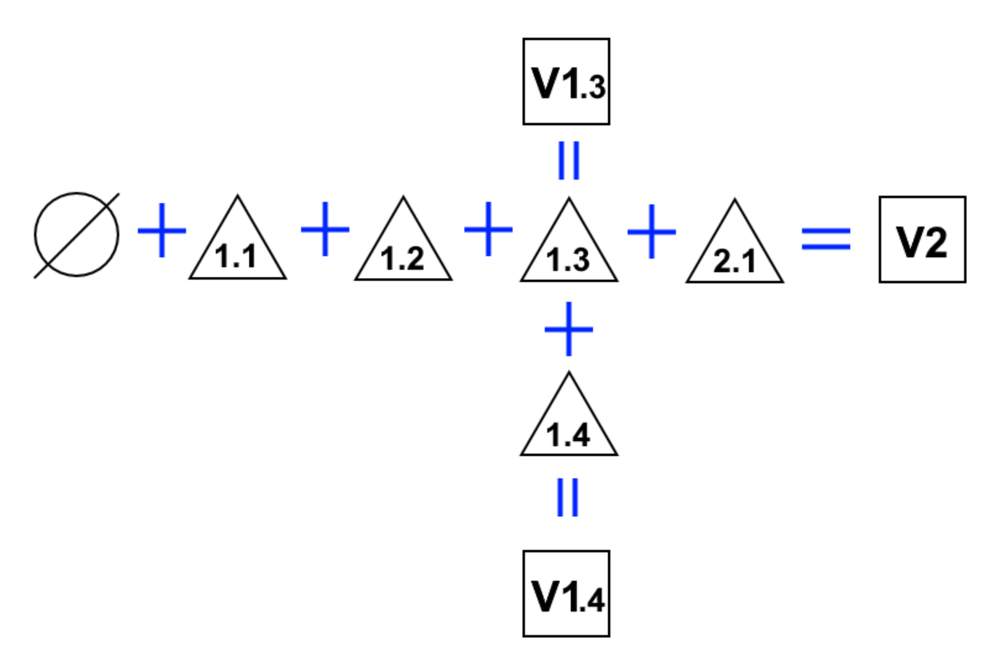


Рис. 1.1

Хранение файла в виде последовательности изменений: начиная с пустого файла в результате трёх шагов получена первая версия программы V1. Дополнительный шаг разработки относится ко второй версии V2. Для исправления ошибки, обнаруженной в версии V1, был выполнен четвёртый шаг, относящийся только к этой версии и не влияющий на версию V2. При необходимости в версии V2 надо произвести аналогичные изменения

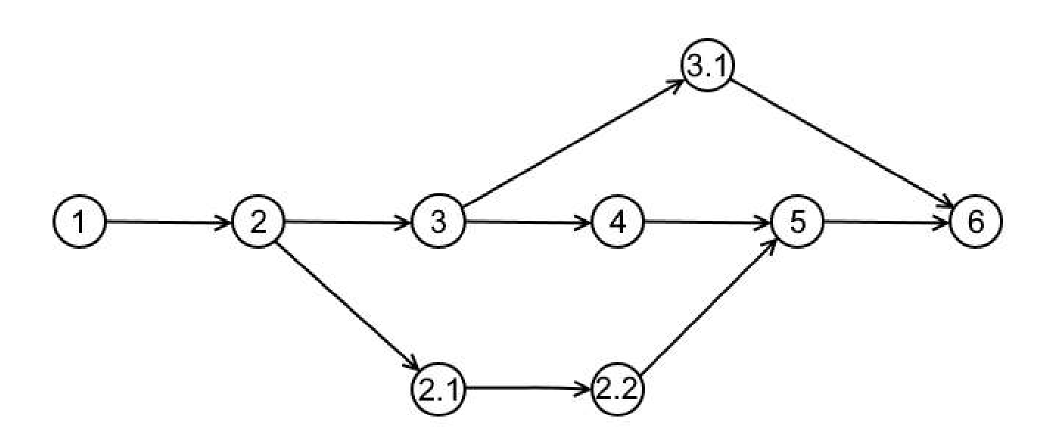


Рис. 1.2

Направленный ацикличный граф истории изменений объекта. В вершинах 2 и 3 произведены ответвления от основной линии, которые затем вливаются в неё в вершинах 5 и 6

В 1982 г. Уолтер Ф. Тичи (Walter F. Tichy) написал статью «Проектирование, реализация и исследование системы контроля изменений» о разработанном в американском университете Пердью (Purdue University) программном продукте RCS (Revision Control System). Этот продукт, состоящий в соответствии с идеологией UNIX из нескольких простых утилит, позволяет:

- помещать файл в его текущем состоянии в хранилище, снабжая его отметкой времени, номером версии, информацией об авторе и кратким описанием произведенных изменений;
- извлекать файл требуемой версии из хранилища для просмотра или внесения изменений;
- создавать ответвление от генеральной линии разработки для исправления ошибок или апробирования новаторских идей;
- внедрять изменения, произведенные в ответвлениях, в основную ветвь разработки;
- просматривать историю изменений.

Особой гордостью разработчиков стала эффективная реализация хранения версий текстовых файлов — отдельными ретроспективными блоками отличий (reverse separate deltas), что увеличивает скорость работы системы при незначительном увеличении объема памяти, необходимой для хранения истории изменений. Этот подход основан на том факте, что последние версии файлов требуются гораздо чаще, чем предыдущие. Поэтому с точки зрения производительности выгоднее хранить актуальные варианты файлов в явном виде, а предыдущие версии получать путём применения к ним обратных изменений.

В системе RCS используется уже привычная терминология:

- check-in — запись в хранилище;
- check-out — извлечение из хранилища;
- fork и branch — разветвление и ответвление;
- merge — слияние.

В локальных системах контроля версий основным методом решения потенциального конфликта изменений, одновременно внесённых разными разработчиками в один и тот же файл, является его предотвращение с помощью механизма блокировок. То есть файл, взятый из системы контроля версий для модификации одним из участников проекта, становится недоступным для остальных членов команды до тех пор, пока он не будет возвращён обратно в систему.

1.4. Централизованные системы контроля версий

Локальные системы контроля версий, рассмотренные выше, помогают организовать работу специалиста, единолично решающего поставленную перед ним задачу, или коллектива сотрудников, использующих общее файловое хранилище. Последний вариант был актуален в период превалирования больших и малых ЭВМ, которые являлись ядром вычислительного комплекса и центром хранения и обработки данных. Коллективный доступ для пользователей к таким ЭВМ обеспечивался посредством дисплейных терминалов.

С распространением в начале 1980-х гг. персональных компьютеров наблюдаются две тенденции. С одной стороны, каждый пользователь становится обладателем вычислительных ресурсов, достаточных для автономной работы. Теперь не требуется записываться в график использования машинного времени и идти на работу в дисплейный зал. С другой стороны, решаемые задачи требуют взаимодействия с такими же автономными единицами и регулярного обмена своими наработками.

Набирает популярность идея объединения персональных компьютеров в вычислительные сети, но постоянное подключение скорее является исключением, чем правилом. В большинстве случаев работа в сети происходит короткими сеансами, для чего пользователь подключается к ней с помощью модема через обычную телефонную линию. Распространённым средством хранения и переноса информации является гибкий магнитный диск.

В таких условиях родилась система параллельных версий CVS (Concurrent Versions System), которую Дик Грюн (Dick Grune), доктор наук Амстердамского свободного университета, в своей статье назвал способом независимого сотрудничества. Эта система являлась надстройкой над системой RCS и представляла собой набор из 25 сценариев оболочки UNIX. Она была успешно использована в проекте распределённой разработки компилятора ACK (Amsterdam Compiler Kit).

При разработке системы CVS основное внимание уделялось решению проблемы одновременной модификации одного и того же файла разными пользователями — конфликта изменений. Для этого была введена абстракция репозитория — единого хранилища всех версий файлов проекта. Доступ к этому хранилищу осуществлялся посредством набора специальных утилит.

Работа над проектом с точки зрения системы CVS заключается в периодическом выполнении двух основных операций:

- получение из репозитория изменений, выполненных другими участниками проекта, и наложение их на свои экземпляры файлов;
- передача в общий репозиторий своих собственных изменений (для этой операции вводится термин commit).

Главное достоинство такой схемы в том, что она позволяет своевременно обнаружить конфликт изменений (рис. 1.3).

Ориентация не на предотвращение конфликтов путём блокировки находящихся в работе файлов, как это было сделано в локальных системах контроля версий, а на их раннее обнаружение и устранение дала возможность применять CVS в распределённой разработке. В ходе работы у каждого участника проекта хранится своя локальная копия необходимых ему файлов. Эти файлы можно модифицировать автономно, без постоянного сетевого подключения. По мере готовности произведенные в файлах изменения передаются в общий репозиторий, по сети или на машинном носителе информации. В этот момент осуществляется их проверка на наличие конфликта с изменениями, произведенными другими сотрудниками. Если обнаружен конфликт, то предлагается его устранить, после чего процедура повторяется.

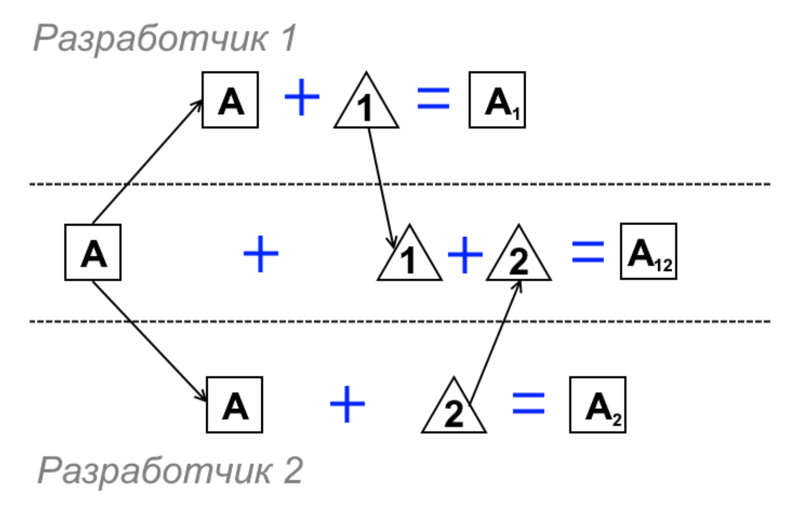


Рис. 1.3

Два разработчика получают из репозитория исходный вариант файлов проекта A, вносят каждый свои изменения D1 и D2, после чего передают их обратно в репозиторий. Результатом наложения изменений является вариант A12. Но изменения могут поступить в другом порядке: сначала D2, а потом D1, в результате чего получится вариант A21. Если A12 отличается от A21, то изменения D1 и D2 конфликтуют между собой

Подход, реализованный в CVS, оказался настолько удачным, что эта система контроля версий стала стандартом де-факто в сфере разработки программного обеспечения и применялась во множестве проектов вплоть до конца XX в. А тем временем развивались сетевые технологии, в начале 1990-х гг. появилась всемирная паутина. Компьютеризация уверенно вышла за рамки университетов и специализированных учреждений, охватила все сферы жизни.

Стремительными темпами растёт число компаний, предоставляющих ИТ-услуги. Разработка программ приобретает глобальный характер, набирает силу движение Open Source, локомотивом которого становится операционная система Linux. Наряду с ростом сложности программ возрастают требования к их надёжности. Для повышения их качества проводятся мероприятия по улучшению процесса разработки, внедряются системы управления заявками на доработку, системы автоматизации сборки и тестирования.

В конце 1990-х гг. приходит понимание, что необходим перевод удачных идей, реализованных в CVS, на новую технологическую платформу. Заказ на решение этой задачи поступил от производителя систем для совместной работы CollabNet в начале 2000-х гг. Целью проекта, который получил название Subversion (иногда используется сокращение SVN), была не разработка революционной системы контроля версий, а решение проблем, которые выявились в CVS за годы промышленной эксплуатации и в современном программном окружении. Уже к сентябрю 2001 г. степень готовности системы Subversion достигла такого уровня, который позволил ей вести её же собственный репозиторий (до этого разработчиками использовалась CVS).

Ключевыми отличиями Subvesrion от CVS стали:

- ведение контроля не только над файлами, но и над каталогами;
- атомарность (транзакционность) операции записи в репозиторий совокупности изменений, выполненных в различных файлах проекта;
- использование алгоритма определения различий на двоичном уровне, а не на уровне строк текстов;
- открытая архитектура, обеспеченная реализацией API (интерфейса программного взаимодействия).

Благодаря модульной архитектуре (рис. 1.4) ядро системы контроля версий Subversion может:

- использовать различные хранилища для репозитория — систему управления базами данных или традиционную файловую систему;
- управляться со стороны клиентов различными способами — локально из командной строки или через сеть по протоколам HTTP или SSH.

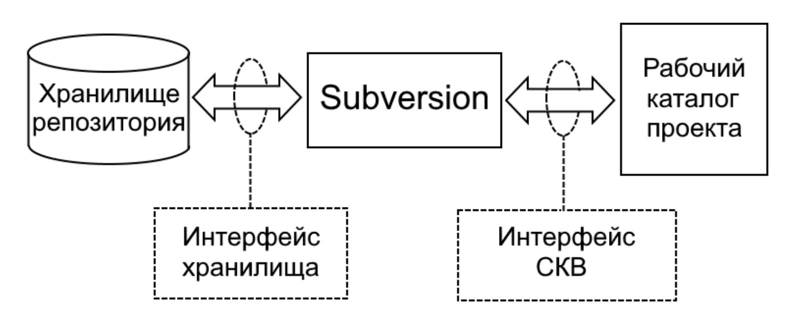


Рис. 1.4

В модульной архитектуре Subversion ядро системы контроля версий (СКВ) отделено от хранилища репозитория и клиентов

Такая открытая реализация системы контроля версий позволяет использовать её в составе интегрированных программных систем для коллективной работы, включающих в себя разнообразные инструменты.

1.5. Распределённые системы контроля версий

В 2005 г. на арену выходит одна из самых популярных на сегодняшний день систем контроля версий — Git. Основное её отличие от системы Subversion в том, что она является не централизованной, а распределённой. В системе Subversion полная история изменений всех файлов проекта хранится только в одном месте — в центральном репозитории, а у участников проекта находятся лишь рабочие копии файлов.

Система Git работает иначе: копию репозитория проекта получает каждый участник. Коллективное взаимодействие может строиться по различным схемам — либо с выделенным сервером, что будет напоминать централизованную модель, либо через распределённую систему репозиториев, организованную с учётом ролей участников проекта. Последний вариант реализован в популярных сервисах GitHub и GitLab (рис. 1.5).

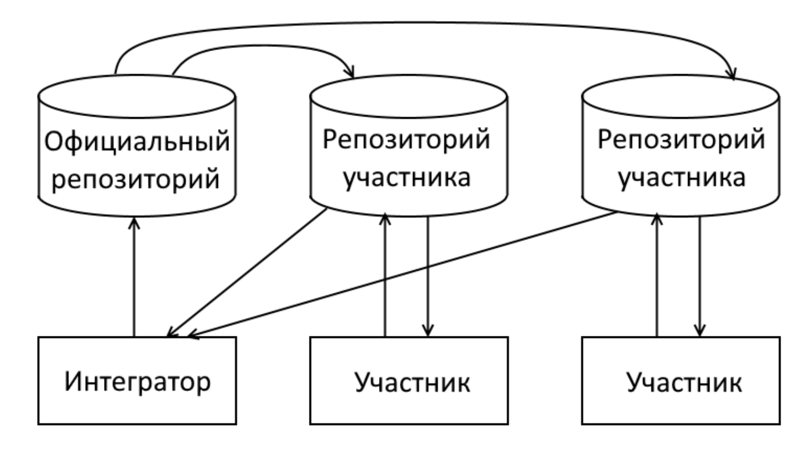


Рис. 1.5

Разработка с распределённой системой контроля версий. Каждый участник работает со своей копией официального репозитория, а интегратор проекта внедряет выполненные ими изменения в официальный репозиторий

Ещё одним отличием Git от ранее рассмотренных систем контроля версий является модель репозитория. Если предшественники смотрели на репозиторий как хранилище истории изменений, выполненных в файлах, то Git трактует репозиторий как виртуальную файловую систему, хранящую снимки рабочего каталога в определённые моменты времени.

Помимо прочего, распределённая архитектура увеличивает скорость выполнения основных операций, в частности помещение моментального снимка рабочего каталога в репозиторий. Если в централизованных системах для этого требуется сетевое подключение, то в Git такая операция выполняется над локальным хранилищем. Синхронизация локального хранилища с сетевым репозиторием сервера — это отдельная фаза использования системы.

Будучи изначально спроектированной для поддержки процесса разработки ядра операционной системы Linux, система Git хорошо подходит для использования в крупных распределённых проектах, хотя пригодна и для локальной разработки.

1.6. О выборе системы контроля версий

Изложенная краткая история развития систем контроля версий иллюстрирует эволюцию представлений о том, как средства автоматизации должны способствовать разработке проектов, в основе которых лежит изменение файлов. Может сложиться впечатление, что на каждом этапе новые системы заменяли своих предшественников, которые безнадёжно устарели. На самом деле это не совсем верно.

История развития вычислительной техники демонстрирует постоянное расширение круга решаемых с помощью компьютеров задач. Сначала это были задачи научных вычислений, затем хозяйственно-экономических расчётов, хра-

нения и обработки алфавитно-цифровой информации, контроля технологического оборудования и управления производственными процессами. Возможности современной вычислительной техники позволяют выполнять обработку мультимедийных данных в реальном времени, обеспечивают их оперативную передачу и хранение. Это привело к созданию на её основе принципиально новых средств общения.

Но с компьютеризацией новых сфер жизни людей старые задачи не исчезают — по-прежнему существует необходимость и в научных вычислениях, и в автоматизации бухгалтерского учёта. И средства, которыми решались эти задачи раньше, даже с позиции современности оказались не так уж плохи. Об этом, к примеру, свидетельствует спрос на специалистов, владеющих языком программирования COBOL, для сопровождения функционирующих систем автоматизации предприятий.

Системы контроля версий сопровождают развитие вычислительной техники и расширение сферы её применения. Все описанные системы были предназначены для решения конкретных задач в определённых условиях, и нельзя сказать, что одна из них определённо лучше других. При выборе системы контроля версий для конкретного проекта надо ориентироваться на его специфические требования.

Например, для индивидуальной разработки даже в современных условиях вполне может подойти система RCS, последняя версия 5.9.4 которой вышла в 2015 г. И уж точно не стоит списывать со счетов систему Subversion, которая находится в активной разработке под эгидой Apache Software Foundation и недавно отметила своё двадцатилетие.

При выборе системы контроля версий для практического применения на первый план могут выйти критерии, о которых ещё не было сказано. Например, в настоящее время нередки случаи, когда коммерческой разработкой программного обеспечения занимаются сотни или даже тысячи программистов, находящихся в разных уголках земного шара, а совокупный размер созданного ими программного кода превосходит возможности хранения, имеющиеся у персональных компьютеров. Кроме того, этот код, являющийся интеллектуальной собственностью компаний, имеет значительную стоимость в денежном выражении. Не удивительно, что в целях предотвращения промышленного шпионажа у них возникает желание ограничить доступ сотрудников только теми участками кода, над которыми они непосредственно работают.

Для удовлетворения противоречивых требований приходится использовать вспомогательные модули для систем контроля версий, которые были рассмотрены выше, или искать альтернативные продукты, коих на сегодняшний день имеется немало — как коммерческих, так и бесплатных.

1.7. Интегрированная система Fossil

Эта книга посвящена бесплатной современной системе управления версиями Fossil, которая была разработана специально для поддержки проекта SQLite, уникального в своём роде и заслуживающего отдельного освещения.

В стандартной классификации Fossil — это распределённая система контроля версий, т. е. относится к тому же классу, что и Git.

Каждый участник проекта, поддерживаемого Fossil, обладает полной копией всей истории изменений файлов проекта. Но модель разработки напоминает централизованную схему, потому что обмен результатами работы между участниками происходит через сетевой репозиторий, размещённый на сервере. В целом рабочий процесс с использованием Fossil выглядит так (рис. 1.6).

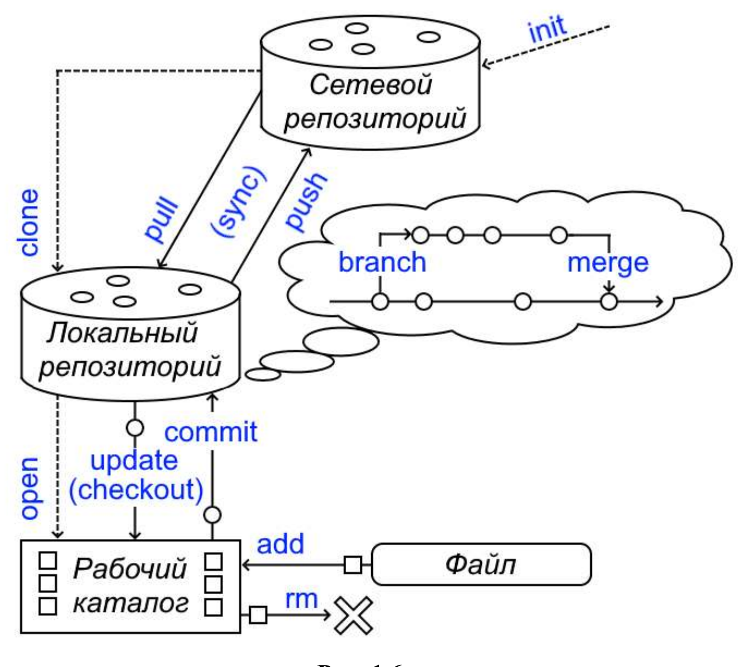


Рис. 1.6

Рабочий процесс с использованием Fossil:

пунктирной линией показаны первоначальные действия, выполняемые однократно при развёртывании системы, сплошной — регулярно выполняемые действия в ходе работы над проектом. Подписи линий отражают действия, ими обозначаемые, и соответствуют командам системы управления версий Fossil.

Развёртывание системы начинается с создания сетевого репозитория (init), который затем копируется участниками проекта на их персональные компьютеры (clone). После этого у них появляется возможность создать рабочие каталоги (open).

Файлы из рабочего каталога могут добавляться в список контролируемых системой Fossil (add), и тогда производимые в них изменения будут отслеживаться, а сами они будут включаться в моментальные снимки рабочего каталога, записываемые в точки сохранения репозитория (commit). При необходимости можно снять контроль с файлов (rm), в результате чего они не будут присутствовать в последующих моментальных снимках, но навсегда останутся в тех, которые были сделаны ранее.

Чтобы результат операций, выполненных над рабочим каталогом и локальным репозиторием одного участника проекта, стал доступен остальным участникам, каждый должен синхронизировать (sync) свой локальный репозиторий с сетевым репозиторием. Процесс синхронизации заключается в получении в локальный репозиторий отсутствующих в нём изменений из сетевого репозитория (pull) и отправке изменений, выполненных участником после предыдущей синхронизации, в сетевой репозиторий (push).

Может оказаться, что пока разработчик модифицировал имевшийся у него на компьютере вариант файла, внесённые им изменения вошли в противоречие с версией, появившейся к этому времени в сетевом репозитории. Тогда при синхронизации репозиториев системой контроля версий это противоречие будет обнаружено, разработчик получит сообщение о конфликте, и синхронизация будет отложена до устранения конфликта.

Лучший вариант решения конфликта — ознакомиться с наложившимися исправлениями и сформировать из них правильную комбинацию, после чего завершить синхронизацию репозиториев. Если оперативное решение конфликта по каким-то причинам невозможно, то существует вариант его эскалации — основная линия разработки проекта разветвляется (fork), в результате чего возникает ответвление (branch), в котором файл, явившийся причиной конфликта, получит только те изменения, которые произвёл разработчик. А вариант файла с конфликтующими изменениями, имеющийся в сетевом репозитории, останется в основной ветке проекта.

С точки зрения развития проекта его ветви, появившиеся в результате неразрешённых конфликтов, надо как можно скорее приводить к общему знаменателю и объединять (merge) с основной ветвью. В то же время наличие параллельных ветвей разработки для тестирования новых функций, исправления ошибок или сопровождения конкретных выпусков продукта проблемой не является. Возможность в произвольный момент переключиться на любой из ранее созданных моментальных снимков рабочего каталога (checkout) и внести в него необходимые изменения является одной из основных причин использования систем контроля версий.

В целом описанная выше методика характерна для многих современных систем контроля версий. Но у Fossil есть несколько особенностей, которые выделяют её на фоне конкурентов.

Во-первых, в качестве хранилища своих данных Fossil использует локальную СУБД SQLite — очень надёжное, совместимое и легко переносимое на различные платформы решение. Это упрощает администрирование системы: все данные репозитория сосредоточены в одном файле, а сведения о состоянии рабочего каталога — в другом. Файлы формата SQLite без изменений могут быть перенесены на другую платформу и продолжат работать там без каких бы то ни было проблем.

Во-вторых, в Fossil помимо собственно системы контроля версий входит набор подсистем, реализующих полезные в работе над проектом функции:

- управление заявками на доработку (Bug Tracking);
- документирование (Wiki);

- обсуждение (Forum);
- веб-сервер.

Для использования перечисленных инструментов с другими системами контроля версий требуются подключение и настройка дополнительных модулей или сторонних программных продуктов. В Fossil они реализованы с помощью того же механизма, на котором работает версионированное хранение файлов проекта. Такое решение экономично как с точки зрения повторного использования программного кода, так и с точки зрения эффективного хранения данных этих подсистем. Вся информация находится в одном файле базы данных SQLite, что исключает утрату истории доработок, обсуждения или сопроводительной документации проекта.

Встроенный в Fossil веб-сервер служит как для предоставления доступа к наглядному и функциональному веб-интерфейсу, так и может быть использован для быстрого развёртывания и дальнейшего функционирования статического веб-сайта.

В-третьих, все функции Fossil реализуется одним-единственным исполняемым файлом, который не имеет зависимостей от сторонних библиотек. Это упрощает до минимума как установку системы, так и её сопровождение.

Благодаря перечисленным особенностям система Fossil становится привлекательной для индивидуальных разработчиков, которых в иной ситуации могла бы отпугнуть сложность нового, пусть и полезного, инструмента.

Систему Fossil можно использовать в образовании. На её основе легко создать учебный стенд для демонстрации не только основных положений систем контроля версий, но и смежных инструментов, активно используемых в реальных проектах.

На эту систему стоит обратить внимание и организациям, в коллективах которых по каким-то причинам система контроля версий ещё не применяется. По сравнению с современными альтернативами у неё может и не быть явных преимуществ, но какая-нибудь из перечисленных особенностей может оказаться полезной.

Взаимодействие пользователя с системой Fossil осуществляется двумя способами: посредством командной строки и через веб-интерфейс. Из командной строки, открытой на рабочем каталоге проекта, можно быстро записать состояние рабочего каталога в репозиторий и запросить обновления. Через веб-интерфейс удобно производить настройку системы, просматривать шкалу времени проекта и статистику, участвовать в обсуждениях на встроенном форуме (рис. 1.7).

Основные элементы веб-интерфейса системы Fossil показаны на рисунке 1.8. В самом верху клиентской области окна веб-браузера отображается строка заголовка (1). Она содержит название открытого репозитория и обозначение текущего режима работы, которые разделены наклонной чертой. В правом верхнем углу находится гиперссылка Login (2), с помощью которой пользователь может авторизоваться в системе и получить доступ к выполнению разрешённых ему операций. После авторизации там будут отображаться имя учётной записи активного пользователя и гиперссылка для выхода из системы Logout.

Рис. 1.7

Веб-интерфейс системы Fossil в режиме отображения статистики работы над проектом

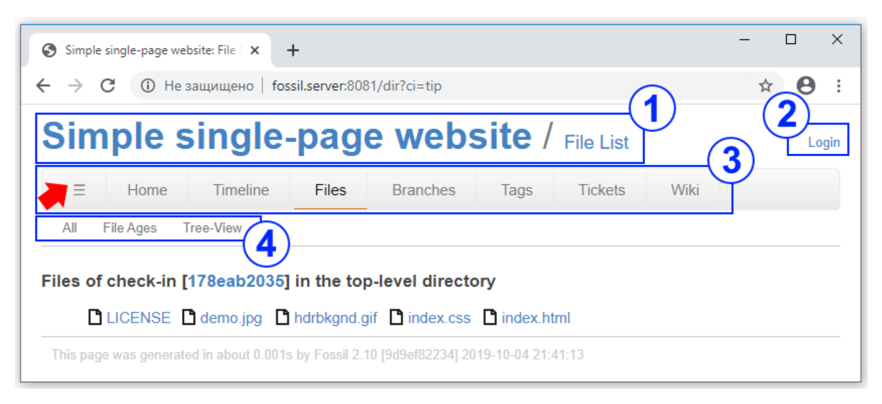


Рис. 1.8

Основные элементы веб-интерфейса системы Fossil:

1 — заголовок; *2* — гиперссылка для авторизации в системе; *3* — строка главного горизонтального меню; *4* — строка дополнительного горизонтального меню. Стрелкой обозначен пункт вызова раскрывающегося меню.

Под строкой заголовка находится главное горизонтальное меню (3) системы Fossil, в котором можно выбрать желаемый режим работы. В этом меню представлены только основные возможности, кроме того, состав пунктов меню может сокращаться при уменьшении ширины окна веб-браузера. Первым в главном меню идёт пункт вызова раскрывающегося меню (на рисунке обозначен жирной стрелкой), в котором представлен исчерпывающий список режимов работы, разрешённых текущему пользователю. Некоторые режимы работы Fossil предоставляют пользователю дополнительные возможности, выбор которых осуществляется в дополнительном горизонтальном меню (4).

При использовании командной строки в окне терминала операционной системы вводятся с клавиатуры команды в формате:

```
fossil команда параметры
```

Например, наиболее часто используемая в ходе работы над проектом команда загрузки в репозиторий состояния рабочего каталога выглядит следующим образом:

```
fossil commit -m "Описание выполненной работы"
```

Несмотря на то, что некоторые операции можно выполнить как через веб-интерфейс, так и из командной строки, перечисленные способы работы с Fossil являются взаимодополняющими. При отсутствии практики работы в режиме командной строки первоначально может ощущаться дискомфорт, но ввод команд с клавиатуры быстро входит в привычку.

Глава 2
УСТАНОВКА FOSSIL

2.1. Получение исполняемого файла

Исчерпывающее описание установки Fossil можно уместить в один абзац. Вся функциональность этой системы заключена в единственном исполняемом файле. Поэтому для её установки надо загрузить с официального сайта https://fossil-scm.org/ (рис. 2.1) архив с исполняемым файлом, извлечь этот файл и разместить в каталоге, который включён в список путей поиска программ по умолчанию.

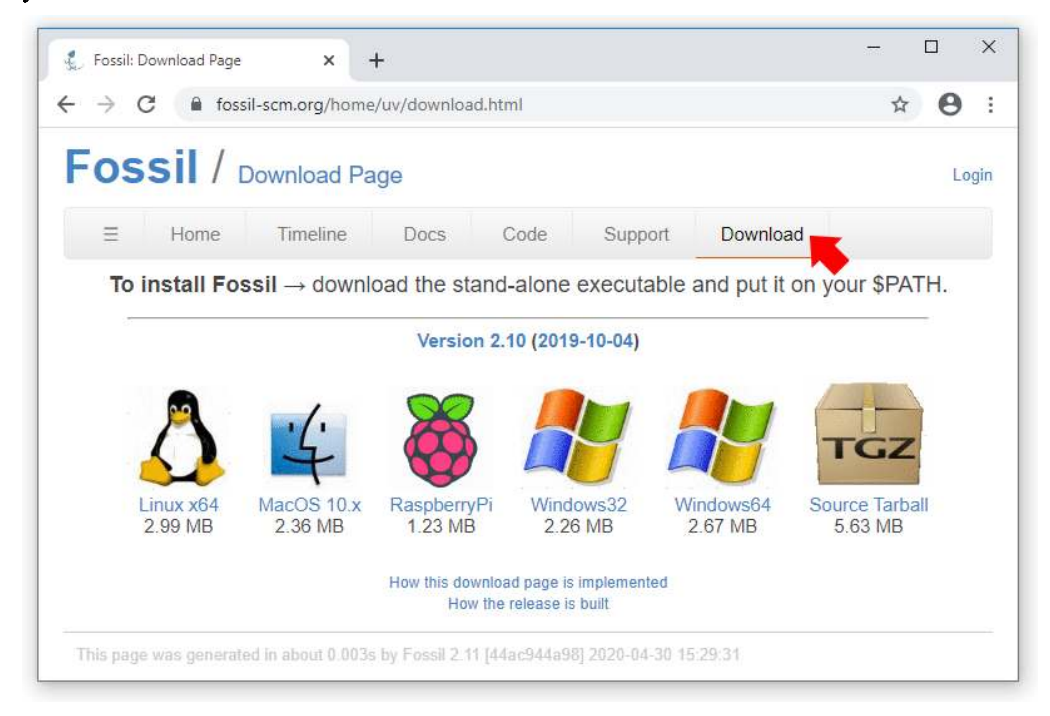


Рис. 2.1

Страница загрузки исполняемых файлов и исходных текстов системы Fossil

Вероятно, многие читатели этой книги в дополнительных инструкциях не нуждаются. Тем не менее давайте вспомним команды, которые требуется выполнить для установки Fossil в разных операционных системах, рассмотрим пример сборки Fossil из исходных текстов и обсудим вопросы, касающиеся повышения удобства использования Fossil в повседневной работе.

Помимо загрузки исполняемого файла с официального сайта, в популярных дистрибутивах операционной системы GNU/Linux имеется возможность получения его штатными средствами операционной системы из официального репозитория программного обеспечения. Например, чтобы установить Fossil в

операционной системе, основанной на дистрибутиве Ubuntu или Debian, надо в терминале от имени суперпользователя root выполнить команду:

```
apt-get install fossil
```

В дистрибутиве Mageia, который использует пакеты RPM, того же результата можно достичь с помощью команды:

```
urpmi fossil
```

Даже в репозитории операционной системы MINIX 3 имеется двоичный пакет системы Fossil, правда, устаревшей на сегодняшний день версии 1.28. Установить его можно с помощью команды:

```
pkgin install fossil
```

Установка Fossil из репозитория во многих случаях является наиболее предпочтительным вариантом, потому что избавляет от необходимости заниматься вопросом размещения загруженного исполняемого файла в правильный системный каталог. После такой установки программа сразу готова к использованию.

Однако в некоторых ситуациях может потребоваться сборка исполняемого файла из исходных текстов. Пользуясь случаем, рассмотрим выполнение этой процедуры в операционной системе MINIX 3, чтобы получить и для неё самую актуальную на момент написания книги стабильную версию Fossil 2.10.

2.2. Сборка из исходных текстов

Для сборки исполняемого файла Fossil из исходных текстов в среде операционной системы MINIX 3 потребуются инструменты, представленные пакетами binutils и clang. Если они ещё не были установлены, то от имени суперпользователя root надо выполнить команды:

```
pkgin update
pkgin install binutils
pkgin install clang
```

Чтобы конфигуратор Fossil обнаружил установленный компилятор clang, надо создать символическую ссылку:

```
ln -s /usr/pkg/bin/clang /usr/pkg/bin/cc
```

Дальнейшие действия можно выполнять от имени рядового пользователя операционной системы. Прежде всего, потребуется загрузить с официального сайта проекта Fossil исходные тексты устанавливаемой программы с помощью команды:

```
ftp http://fossil-scm.org/home/uv/fossil-src-2.10.tar.gz
```

Здесь нет ошибки, в операционной системе MINIX 3 программа ftp умеет загружать файлы из сети по протоколу HTTP. В результате в текущем каталоге должен появиться файл fossil-src-2.10.tar.gz. Для его распаковки надо ввести команду:

```
tar zxf fossil-src-2.10.tar.gz
```

В текущем каталоге появится подкаталог fossil-2.10. Теперь надо сделать каталог с исходными текстами текущим, выполнить конфигурирование исходных текстов и запустить процесс сборки исполняемого файла:

```
cd fossil-2.10
./configure
make
```

Вероятнее всего, через несколько минут на экране появится сообщение об ошибке:

```
clang: warning: argument unused during compilation: '-g'
bld/piechart.o: In function 'piechart_render':
/home/ussr/fossil-2.10/./src/piechart.c:177: undefined reference
to 'sin'
/home/ussr/fossil-2.10/./src/piechart.c:178: undefined reference
to 'cos'
/home/ussr/fossil-2.10/./src/piechart.c:180: undefined reference
to 'sin'
/home/ussr/fossil-2.10/./src/piechart.c:181: undefined reference
to 'cos'
/home/ussr/fossil-2.10/./src/piechart.c:193: undefined reference
to 'sin'
/home/ussr/fossil-2.10/./src/piechart.c:194: undefined reference
to 'cos'
bld/sqlite3.o: In function 'fts5Bm25GetData':
/home/ussr/fossil-2.10/./src/sqlite3.c:206042: undefined reference
to 'log'
clang: error: linker command failed with exit code 1 (use -v to
see invocation)
```

В нём сообщается о том, что компоновщик не нашёл библиотеку с реализациями функций sin и cos. Для устранения этой ошибки надо открыть в текстовом редакторе файл Makefile, найти строку, в которой устанавливается значение переменной *LIB*, и дописать в ней опцию *-lm*. Строка должна приобрести вид:

```
LIB = -L/usr/pkg/lib -lssl -lcrypto -lz -lm
```

После этого надо сохранить файл, выйти из редактора и повторно ввести команду make. В текущем каталоге появится исполняемый файл fossil. Чтобы установить его в операционную систему, надо от имени привилегированного пользователя выполнить команду:

```
make install
```

Она запишет собранный из исходных текстов исполняемый файл в каталог /usr/local/bin, после чего он станет доступен для использования. Проверить результат установки можно с помощью команды:

```
fossil version
```

Она выведет на экран сообщение о версии Fossil:

```
This is fossil version 2.10 [9d9ef82234] 2019-10-04 21:41:13 UTC
```

Таким образом, самая актуальная стабильная версия Fossil была собрана из исходных текстов и установлена в операционную систему.

2.3. Подготовка к использованию в Windows

Казалось бы, установка Fossil в операционной системе Windows не должна вызывать никаких проблем. Что может быть проще, чем записать один-единственный EXE-файл в каталог, прописанный в переменной окружения PATH? И правда, в прошлых версиях операционной системы это смог бы проделать любой подготовленный пользователь.

В современных версиях Windows всё может оказаться не так просто, как ожидалось. С одной стороны, в каталоги, которые уже прописаны в переменной окружения PATH, запись разрешена только пользователям с правами администратора системы. И это разумно, потому что удерживает от простого, но неправильного решения поместить программу fossil.exe в какой-нибудь из системных каталогов. Это — опрометчивый шаг как с точки зрения безопасности системы, так и с точки зрения поддержания её чистоты. С другой стороны, не так-то просто добраться до окна интерфейса, посредством которого можно изменить перечень каталогов, в которых осуществляется поиск программ по умолчанию.

Ниже описан наиболее рациональный с точки зрения автора способ установки программы fossil.exe, пригодный для использования в любой версии операционной системы Windows.

Во-первых, надо создать отдельный каталог для однофайловых программ. Сейчас туда будет помещён файл fossil.exe, но в дальнейшем в него же можно записывать другие полезные утилиты. Имя каталога должно быть максимально простым (содержать только латинские буквы) и коротким, а также находиться максимально близко к корню файловой системы, чтобы исключить как трудности его включения в переменную PATH, так и проблемы работы запущенных из него программ. Хороший вариант имени — EXE, а местоположения — корень системного диска. Итак, создаём каталог C:\EXE и записываем в него программу fossil.exe.

Теперь путь C:\EXE надо добавить в переменную окружения PATH. Откроем окно командной строки Windows (для этого надо нажать комбинацию клавиш WinKey + R, набрать cmd.exe в открывшемся окне, после чего нажать кнопку Ок (рис. 2.2).

Затем в окне командной строки надо ввести команду sysdm.cpl, после чего система выведет окно системы контроля учётных записей, в котором надо разрешить работу оснастки нажатием кнопки Да. В окне оснастки *Свойства системы* (рис. 2.3), которое откроется после этого, надо выбрать вкладку *Дополнительно* и нажать кнопку Переменные среды...

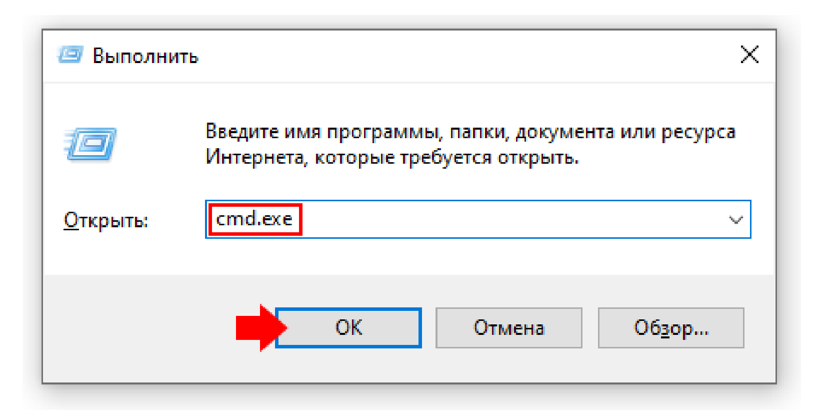


Рис. 2.2

Вызов окна командной строки Windows

Свойства системы

Имя компьютера | Оборудование

Дополнительно | Защита системы | Удаленный доступ

Необходимо иметь права администратора для изменения большинства перечисленных параметров.

Быстродействие

Визуальные эффекты, использование процессора, оперативной и виртуальной памяти

Параметры...

Профили пользователей

Параметры рабочего стола, относящиеся ко входу в систему

Параметры...

Загрузка и восстановление

Загрузка и восстановление системы, отладочная информация

Параметры...

Переменные среды...

ОК | Отмена | Применить

Рис. 2.3

Окно оснастки Свойства системы

Откроется окно редактирования переменных среды, состоящее из двух частей (рис. 2.4). В верхней части находятся переменные, действующие только для текущего пользователя операционной системы, в приведенном примере это PCUser. В нижней части находятся переменные, которые действуют для всех пользователей операционной системы.

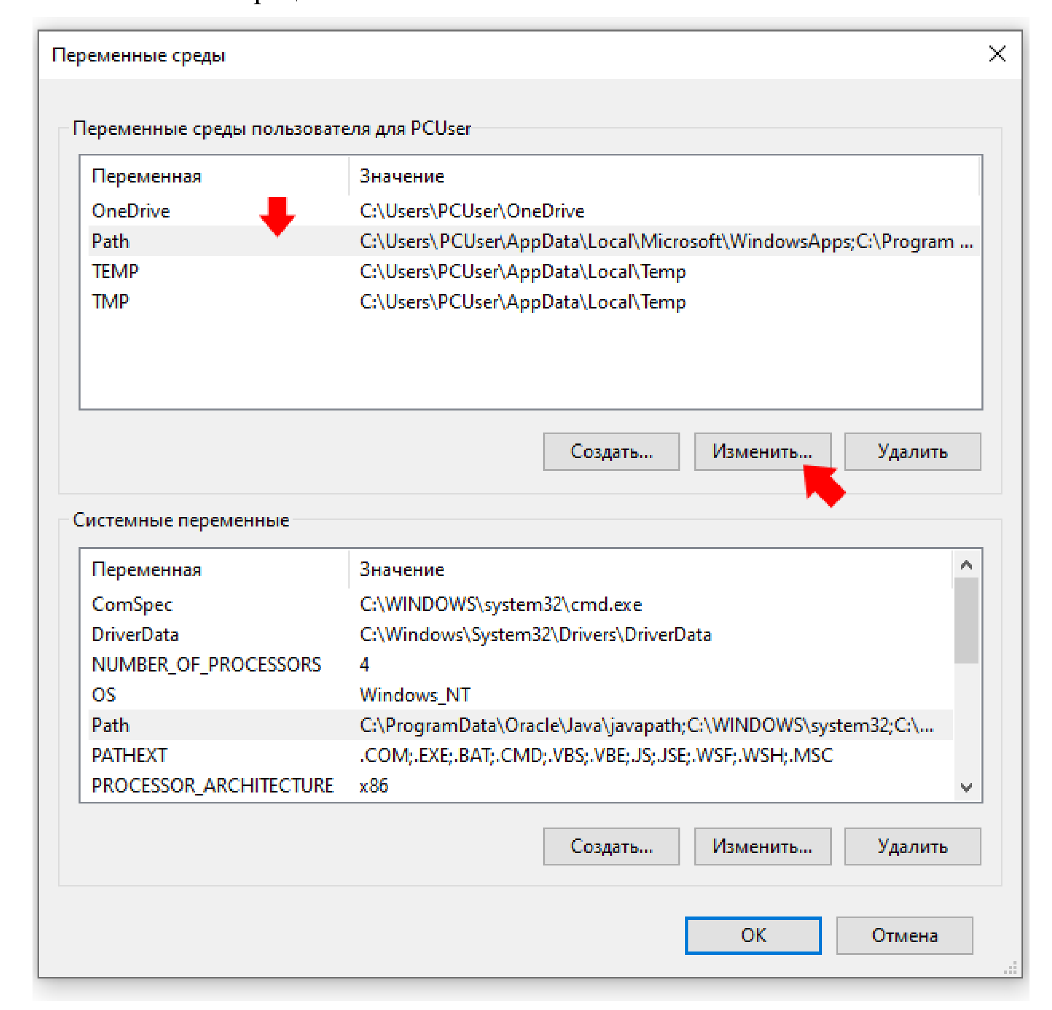


Рис. 2.4

Окно редактирования переменных среды

Исправим значение переменной среды окружения PATH для текущего пользователя. Для этого надо в списке переменных выбрать строку, начинающуюся с Path, после чего нажать кнопку Изменить… Откроется окно Изменить переменную среды (рис. 2.5).

В открывшемся окне надо нажать кнопку Создать. Это активирует строку редактирования, позволяющую добавить новый путь к списку существующих путей переменной среды окружения PATH. В строке редактирования надо набрать C:\EXE, после чего нажать кнопку OK в окне изменения переменных среды.

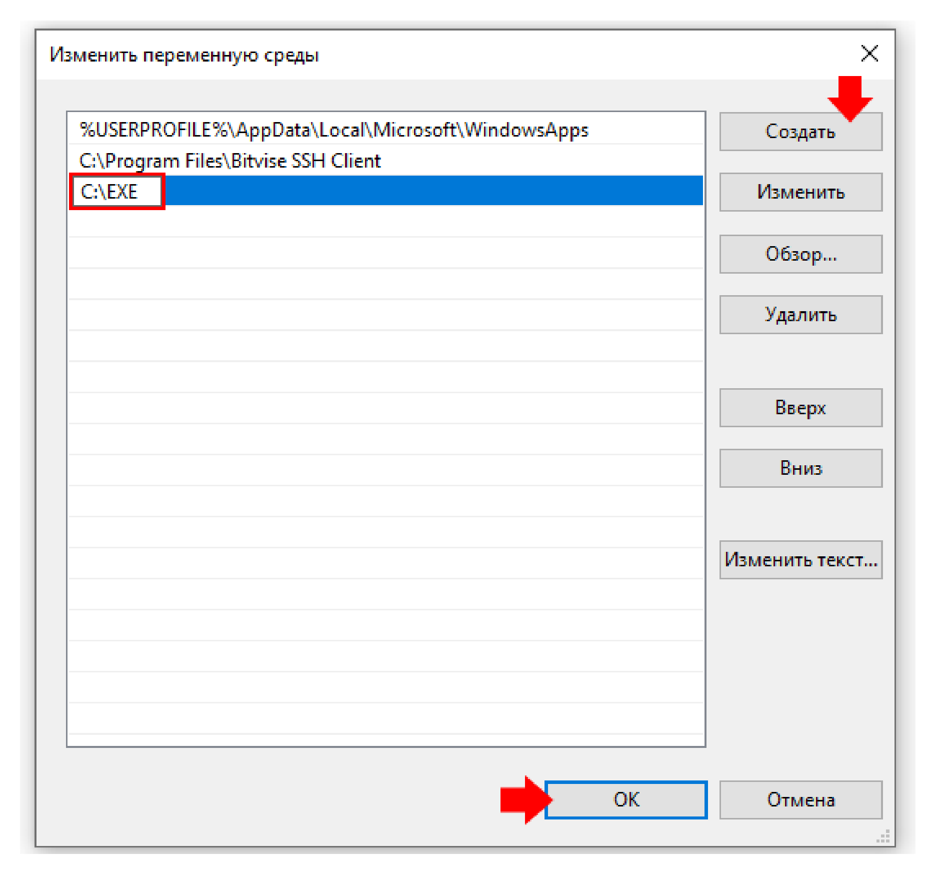


Рис. 2.5

Окно внесения изменений в переменные среды

Задача выполнена — путь к новому каталогу для исполняемых файлов добавлен. Оставшиеся на экране диалоговые окна можно закрыть нажатиями на кнопку OK. Осталось записать в каталог C:\EXE программу fossil.exe, если это ещё не было сделано. Теперь операционная система будет находить программу при указании только её имени, без необходимости задания полного пути к её местонахождению.

2.4. Альтернативный способ для Windows

Если изменять значения переменных окружения для всего сеанса работы операционной системы нежелательно или такая возможность отсутствует в силу установленных администратором политик, можно организовать работу с системой контроля версий иначе. Возможно, кто-то даже увидит в предлагаемом варианте преимущества по сравнению с тем, который был рассмотрен ранее.

Система контроля версий управляется из командной строки, в которой текущим должен быть рабочий каталог проекта, над которым ведётся работа. Можно совместить действия по запуску сеанса командной строки, установке текущего рабочего каталога и указанию пути к программе fossil.exe. Далее пред-

полагается, что программа fossil.exe размещена в каталоге C:\EXE, но настройка переменной окружения PATH не производилась.

Для изменения значения переменной среды окружения PATH в каталоге C:\EXE надо создать командный файл fossil-env.bat со следующим содержимым:

Листинг командного файла C:\EXE\fossil-env.bat

```
@ECHO OFF
CHCP 65001
PATH %PATH%;C:\EXE
ECHO Система контроля версий готова к работе.
```

В первой строке командного файла отключается вывод на экран самих выполняемых команд. Вторая строка устанавливает кодовую страницу номер 65001, что соответствует кодировке UNICODE (по умолчанию командная строка Windows использует кодировку CP-866). Это обеспечит нормальную передачу в систему контроля версий описаний изменений на русском языке, в противном случае при просмотре информации через веб-интерфейс вместо них будут отображаться спецсимволы. В третьей строке к содержимому переменной окружения *PATH* добавляется путь C:\EXE. И, наконец, последняя строка просто выводит на экран информационное сообщение.

Использовать командный файл fossil-env.bat надо с помощью ярлыка, который создаётся следующим образом. Щелчком правой кнопкой мыши на свободном пространстве рабочего стола вызывается контекстное меню, в котором выбирается пункт Создать – Ярлык (рис. 2.6).

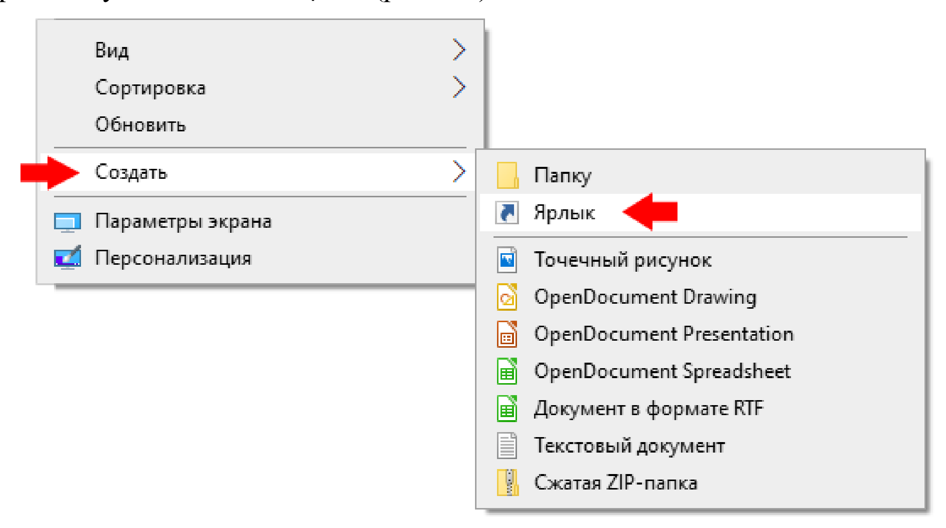


Рис. 2.6

Вызов диалогового окна для создания ярлыка

В поле ввода Укажите расположение объекта появившегося диалогового окна Создать ярлык надо набрать текст команды (рис. 2.7):

```
%COMSPEC% /K C:\EXE\fossil-env.bat
```

Создать ярлык

Для какого элемента нужно создать ярлык?

Этот мастер помогает создать ярлык, указывающий на локальные или сетевые программы, файлы, папки, компьютеры или адреса в Интернете.

Укажите расположение объекта:

%COMSPEC% /K C:\EXE\fossil-env.bat

Обзор...

Для продолжения нажмите кнопку "Далее".

Далее Отмена

Рис. 2.7

Командная строка создаваемого ярлыка

При выполнении этой команды на место *%COMSPEC%* будет подставлен полный путь к командному интерпретатору cmd.exe, параметр */K* означает, что командный интерпретатор не должен завершать свою работу после выполнения командного файла, полный путь к которому указан после этого параметра.

После нажатия кнопки Далее будет предложено указать название ярлыка. Хорошее решение — вписать в это поле краткое название проекта, над которым будет вестись работа (рис. 2.8).

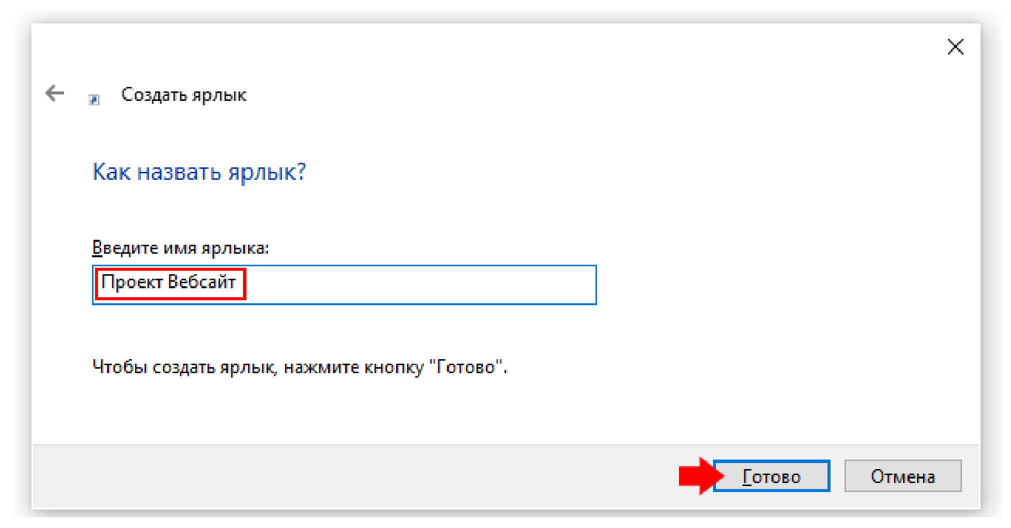


Рис. 2.8

Ввод названия ярлыка

После нажатия кнопки Готово на рабочем столе появится созданный ярлык. Осталось лишь немного его подкорректировать. Надо с помощью правой кнопки мыши вызвать на этом ярлыке контекстное меню, выбрать в нём пункт Свойства. На вкладке Ярлык в поле Рабочая папка: следует указать путь к рабочему каталогу проекта (он должен существовать к этому моменту) и нажать кнопку ОК (рис. 2.9).

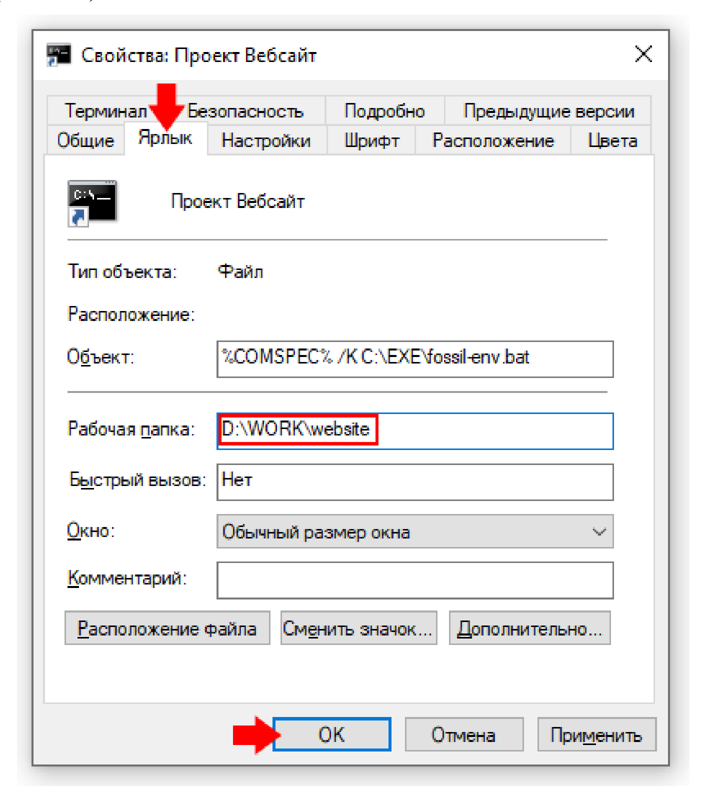


Рис. 2.9

Ввод пути к рабочему каталогу

С помощью созданного ярлыка будет запускаться окно командной строки с кодировкой UNICODE, в котором текущим сразу станет рабочий каталог проекта, и путь к программе fossil.exe будет добавлен в переменную окружения *PATH*. Такие ярлыки можно создать для всех текущих проектов и поместить их в отдельную папку на рабочем столе.

Глава 3
ПОДГОТОВКА К РАБОТЕ НАД ПРОЕКТОМ

3.1. Описание проекта

После того, как система Fossil установлена на компьютере и готова к использованию, можно приступать к работе над проектом. Ранее уже было сказано, что систему Fossil может эксплуатировать как индивидуальный разработчик, так и команда из нескольких человек. Первый вариант проще, и рабочий процесс можно описать линейно, в форме рассказа. Второй вариант может предоставить большее разнообразие ситуаций, и описание процесса разработки будет напоминать роман, в котором взаимодействуют несколько персонажей.

Для более полного раскрытия возможностей системы Fossil дальнейшее повествование будет ориентировано на её коллективное использование. Индивидуальные разработчики при чтении могут пропустить эпизоды, в которых описывается настройка сервера и учётных записей пользователей. Но для получения всестороннего представления о системе будет не лишним ознакомиться со всеми рассмотренными аспектами.

Чтобы изложение не превратилось в сухой перевод на русский язык стандартной документации, будем рассматривать пусть и вымышленный, но конкретный пример. Это позволит наглядно обыграть несколько возможных ситуаций и упростит восприятие материала.

Предположим, что перед небольшой командой специалистов, состоящей из менеджера, верстальщиков и дизайнеров, поставлена задача разработки шаблона простого одностраничного веб-сайта в соответствии с представленным на рисунке 3.1 образцом. Исходя из основных требований к продукту: простота, компактность, адаптивность к устройствам просмотра, независимость от сторонних библиотек и лицензионная чистота — выбрана стратегия разработки с ориентацией на веб-стандарты. Решено использовать чистые HTML, CSS и JavaScript.

Поскольку эта книга не является учебником по языкам веб-разработки, основное внимание в ней сосредоточено не на раскрытии секретов вёрстки или дизайнерских находок, а на организации рабочего процесса, прежде всего — совместного использования исходных текстов проекта.

Подготовка к работе над проектом с использованием системы Fossil включает следующие шаги:

- создание файла репозитория;
- выполнение первичной настройки репозитория;
- размещение репозитория на сервере и предоставление к нему сетевого доступа;
- настройка учётных записей пользователей репозитория.

Рассмотрим их подробно.

О компании Продукция Партнёры Вакансии Приёмные пункты

Рога и копыта натуральный продукт от первого поставщика



О компании

Общество с ограниченной ответственностью «Рога и копыта» основано в 1927 году в городе Черноморске, который славится экологически чистой окружающей средой.



Состав Правления:

Основатель компании — *Остап Ибрагимович Бендер*, известный в профессиональных кругах предприниматель, бизнесмен, инноватор.
Александр Петрович Балаганов — уполномоченный по копытам.
Паниковский Михаил Самуэлевич — директор департамента транспорта и логистики.



Продукция

Предлагаем заботливо собранные и тщательно обработанные рога и копыта самого высокого качества для нужд гребёночной и мундштучной промышленности. С нашими рогами и копытами ваш бизнес достигнет вершин успеха и процветания!

Мы представлены в сети Интернет:

Будем рады сотрудничеству!

Рис. 3.1

Образец простого одностраничного веб-сайта

3.2. Создание репозитория

Каждый Fossil-репозиторий представляет собой отдельный файл базы данных SQLite, в котором хранятся все элементы проекта и вся история его изменений, вплоть до текущего состояния. Внутренняя структура файла Fossil-репозитория одинакова во всех операционных системах и не зависит от архитектуры компьютера.

Файлы Fossil-репозиториев можно копировать с одного компьютера на другой без риска потери их работоспособности. С одной стороны, это открывает простой путь к резервному копированию репозиториев. С другой, нельзя забывать, что в файле копии Fossil-репозитория находится вся имеющаяся в оригинальном репозитории информация, и обладатель такого файла может беспрепятственно получить к ней доступ. Поэтому с точки зрения информационной безопасности доступ к файлам Fossil-репозиториев следует ограничивать.

Рассматриваемый в этой книге проект называется «Простой одностраничный веб-сайт», что может быть переведено на английский язык как «Simple single-page website». Поэтому назовём файл репозитория sspwebsite.fossil. Расширение .fossil говорит о том, что содержимое файла является Fossil-репозиторием, причём не только пользователю, но и веб-серверу Fossil, о чём будет подробнее рассказано в соответствующей главе.

Для создания файла репозитория с именем sspwebsite.fossil надо выполнить команду:

```
fossil init --admin-user SetupUser sspwebsite.fossil
```

Параметр *--admin-user* позволяет указать имя пользователя — создателя и единоличного владельца репозитория, который будет обладать неограниченными правами в отношении созданного репозитория. В приведенном примере это пользователь с именем SetupUser. Если же этот параметр опустить, то по умолчанию владельцу репозитория будет дано имя текущего пользователя операционной системы.

В случае успеха на экран будет выведена информация о созданном репозитории примерно такого вида:

```
project-id: a9dc10379940e886735d86c2a7694947ab3d49e7
server-id: 7e6b78b7e097327027697e0c6c1ffccae8c13c3d
admin-user: SetupUser (initial password is "6787ed")
```

В этих сведениях особый интерес представляет последняя строка, в которой указан начальный пароль пользователя — создателя репозитория, в данном случае 6787ed. Этот пароль требуется для управления репозиторием с другого компьютера по сети, о чём будет сказано позже. А сейчас приступим к первичной настройке созданного репозитория.

3.3. Первичная настройка репозитория

Настройка репозитория выполняется через веб-интерфейс. Доступ к нему проще всего получить, если на том же компьютере, где находится файл репозитория, выполнить команду:

```
fossil ui sspwebsite.fossil
```

В приведённой команде sspwebsite.fossil — это имя файла с репозиторием. Результатом её выполнения будет запуск веб-браузера и открытие в нём домашней страницы проекта (рис. 3.2).

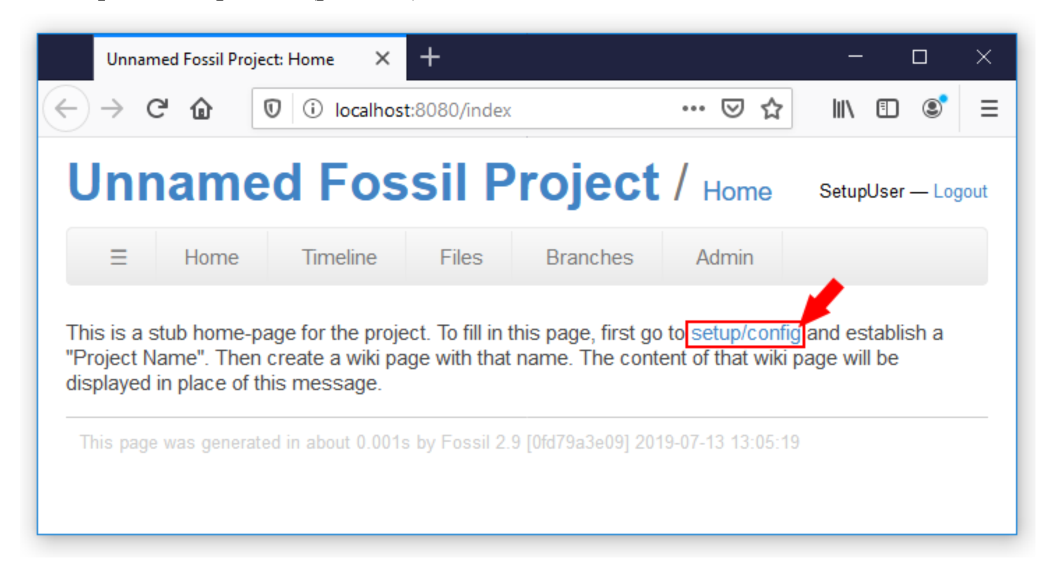


Рис. 3.2

Домашняя страница проекта, открытая в веб-браузере.
Для выполнения настроек надо перейти по гиперссылке setup/config

По факту при выполнении этой команды запускается веб-сервер на TCP-порту 8080 локального сетевого интерфейса, и веб-браузеру предписывается открыть страницу с адресом http://localhost:8080/index. При этом автоматически, без запроса пароля, происходит вход в систему управления репозиторием под именем пользователя-владельца, в данном случае — SetupUser. Это означает, что любой пользователь, получивший непосредственный доступ к компьютеру с файлом репозитория, может выполнить с ним любые действия.

Может оказаться, что порт 8080 на компьютере уже занят каким-нибудь другим сетевым приложением. В этом случае веб-сервер попытается использовать следующий по порядку TCP-порт с номером 8081.

Поскольку веб-сервер, запускаемый для настройки репозитория, привязывается к локальному сетевому интерфейсу localhost, получить к нему доступ с другого компьютера через сеть нельзя. Процедура предоставления сетевого доступа к репозиторию будет описана ниже.

Для первичной настройки репозитория надо перейти по гиперссылке setup/config, в результате чего откроется форма, изображённая на рисунке 3.3.

Достаточно заполнить три верхних поля: Project Name — название проекта, Project Description — описание проекта, и Tarball and ZIP-archive Prefix — префикс tar- и zip-архива.

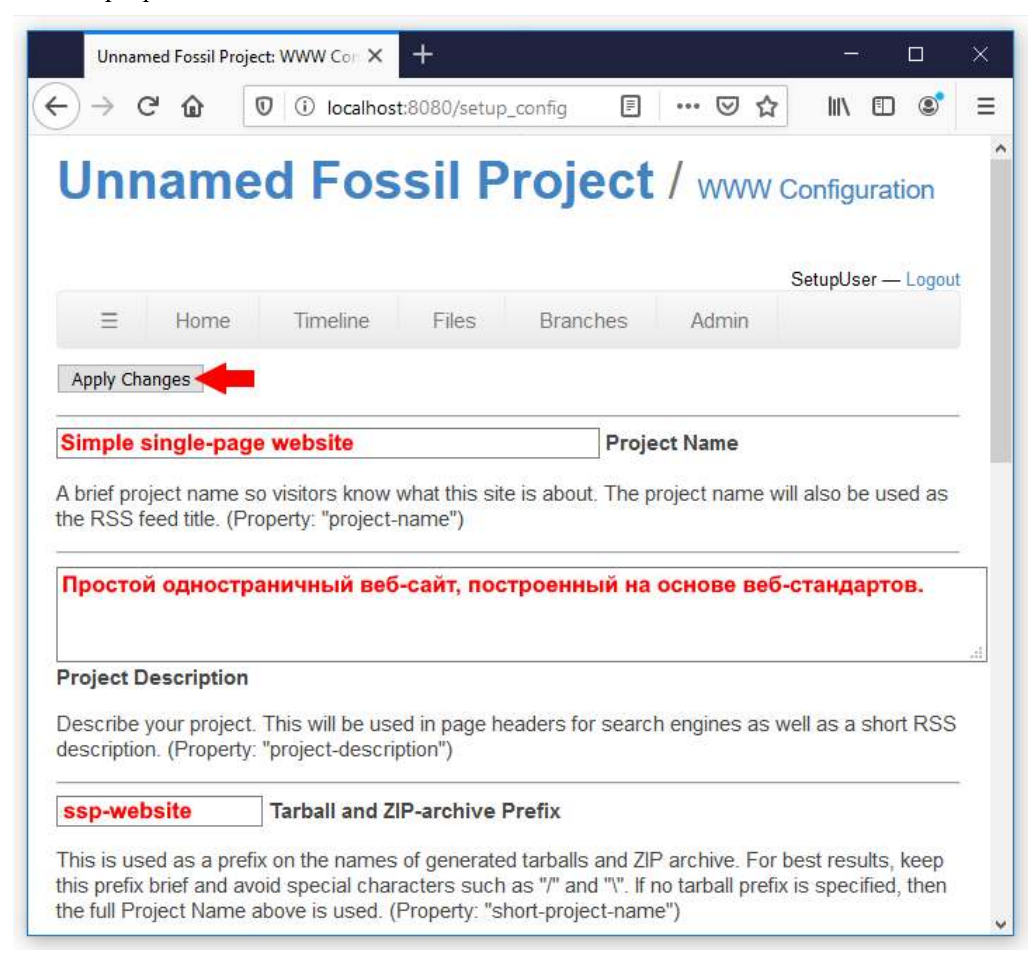


Рис. 3.3

Форма настройки репозитория. После заполнения полей надо нажать кнопку Apply Changes для сохранения изменений

В первое поле вводится название репозитория, которое идентифицирует проект и отображается в заголовке его веб-сайта. Во второе поле вводится описание проекта, которое будет использовано поисковыми системами, если к репозиторию будет открыт интернет-доступ. А в третье поле вводится текст, с которого будут начинаться имена файлов с архивами, содержащими рабочие файлы проекта, при их загрузке с веб-узла.

После заполнения полей для сохранения введённой информации надо нажать кнопку Apply Changes (Применить изменения), которая дублируется вверху и внизу формы. Если в дальнейшем потребуется внести изменения в выполненные настройки, эту форму можно открыть через меню созданного проекта Admin | Configuration (рис. 3.4).

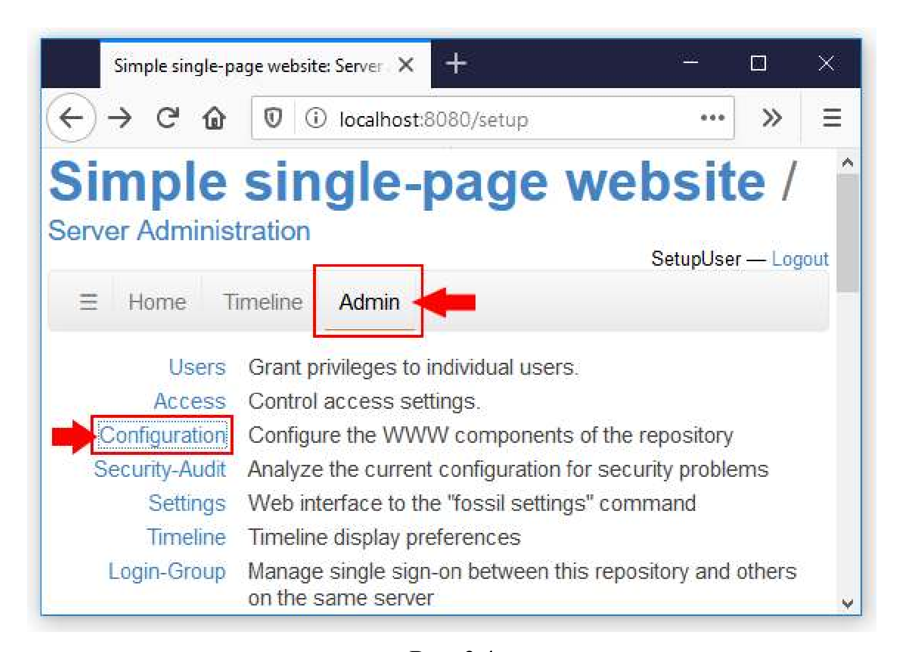


Рис. 3.4

Вызов формы настроек репозитория через меню созданного проекта

3.4. Настройка сетевого доступа

Созданный файл репозитория sspwebsite.fossil можно разместить на сервере и предоставить к нему совместный доступ для коллективной работы над проектом. Для этого нужно на сервере установить программу fossil, переписать на серверный носитель информации файл репозитория (как уже было сказано, это можно сделать простым копированием) и выполнить команду:

```
fossil server -P 8081 sspwebsite.fossil
```

С помощью параметра *-P* указано значение TCP-порта, на котором веб-сервер Fossil будет принимать запросы от клиентов. Его можно не указывать, тогда будет использован порт номер 8080. Об успешном запуске сервера говорит сообщение:

```
Listening for HTTP requests on TCP port 8081
Type Ctrl-C to stop the HTTP server
```

Это же сообщение информирует о том, что для прекращения работы сервера Fossil в этом окне надо нажать комбинацию клавиш Ctrl + C.

В отличие от запуска веб-интерфейса для настройки репозитория с помощью команды fossil ui, которая обсуждалась выше, теперь веб-сервер привязывается не к виртуальному локальному сетевому интерфейсу localhost, а к реальным сетевым интерфейсам компьютера и будет отвечать на любые поступающие из сети запросы. Для проверки его работы можно на другом компьютере запустить веб-браузер и набрать в адресной строке: http://fossil.server:8081/, где fossil.server — сетевое имя сервера. Вместо имени сервера допускается указы-

вать его IP-адрес, узнать который можно с помощью команды ipconfig в операционной системе Windows или команды ifconfig в операционных системах Unix и Linux.

В случае успеха веб-браузер должен отобразить домашнюю страницу репозитория (рис. 3.5). Если страница не открывается, надо внимательно проверить набранный в адресной строке текст, убедиться в правильности указания номера порта (он должен соответствовать тому, который отображается в сообщении сервера). Причиной неработоспособности может быть брандмауэр сервера, блокирующий сетевые подключения.

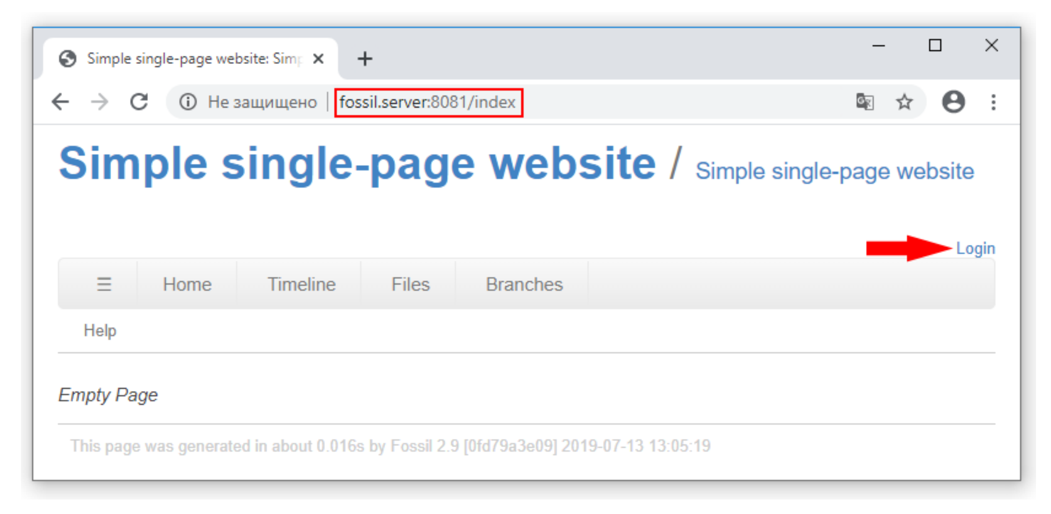


Рис. 3.5

Домашняя страница репозитория, вид из браузера на клиентском компьютере. Для авторизации надо перейти по гиперссылке login

В отличие от рассмотренного ранее случая локального доступа к Fossil, при сетевом подключении не происходит автоматического входа в систему управления репозиторием. Пользователь должен авторизоваться в системе, введя свои имя и пароль в форме, которая открывается при переходе по гиперссылке Login (рис. 3.5).

Для получения полного доступа к управлению репозиторием нужно указать имя пользователя-владельца, использованное при создании файла репозитория, его пароль, который был выдан в результате успешного создания файла, и нажать кнопку Login (Войти). В рассматриваемом примере имя пользователя будет SetupUser, а пароль — 6787ed (рис. 3.6).

Необходимо отметить, что веб-сервер, интегрированный в Fossil, работает по протоколу HTTP, в рамках которого осуществляется обмен сообщениями в открытом виде, без шифрования. Поэтому вся информация, которая передаётся между веб-браузером и сервером, включая регистрационные данные, может быть перехвачена третьей стороной. Это не всегда является проблемой при работе во внутренней сети организации, однако неприемлемо при работе через Интернет.

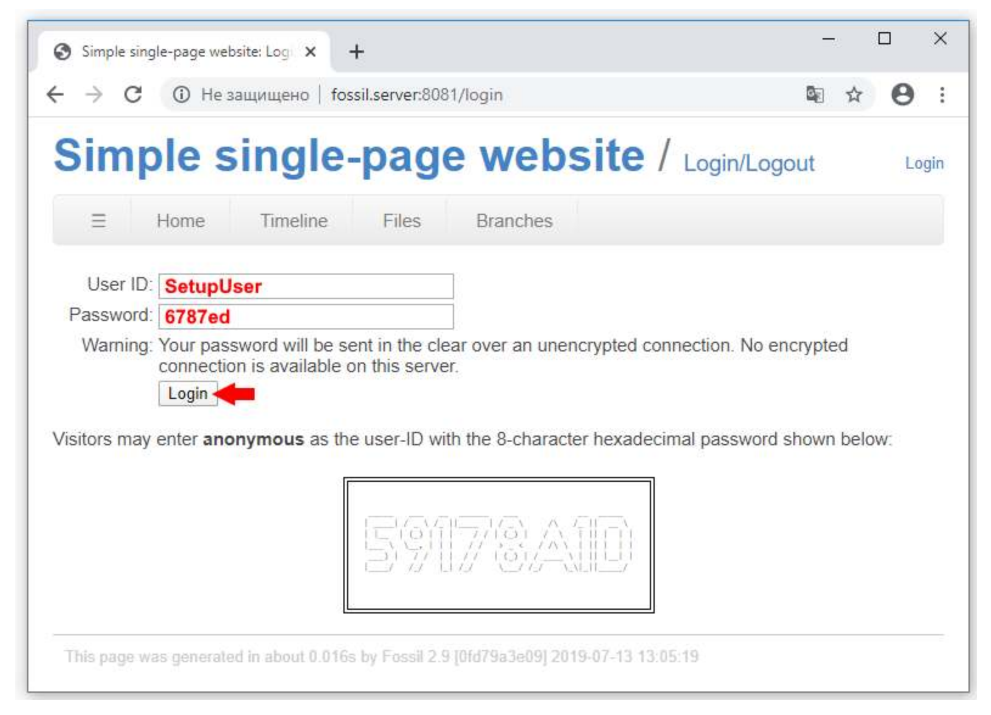


Рис. 3.6

Авторизация в системе управления репозиторием — ввод имени пользователя и пароля

В нижней части окна авторизации, показанного на рисунке 3.6, можно получить сведения, достаточные для авторизации под именем пользователя anonymous. Пароль этого пользователя отображается средствами псевдографики и в рассматриваемом примере является строкой 59178A1D.

Пришло время подробнее поговорить о пользователях Fossil и их правах.

3.5. Совместная работа в информационных системах

Если предполагается эксплуатировать систему Fossil коллективом разработчиков, то для каждого участника проекта необходимо создать персональную учётную запись. Эта процедура очень похожа на предоставление доступа к компьютерам, входящим в состав вычислительной сети.

Когда компьютеры только начали объединять в простейшие сети для совместного использования принтеров и файлов, нередкой была ситуация, при которой все участники разделяли одну и ту же учётную запись, обладающую обычно максимальными правами системного администратора. В небольших коллективах (как правило, полностью сосредоточенных в одном помещении и имеющих возможность непосредственного общения), где люди ответственно относятся к выполняемой работе, обладают достаточной квалификацией и могут подменять друг друга для решения производственных вопросов, такой вариант использования имеет право на существование и по сей день.

Совместное использование информационной системы большим коллективом сотрудников может порождать две задачи. Во-первых, может представлять

интерес, какой именно пользователь совершил в системе те или иные действия. Во-вторых, в соответствии со своими обязанностями пользователи системы должны иметь разный уровень доступа к системе во избежание утечки конфиденциальных сведений и для предотвращения нарушений нормального течения рабочего процесса из-за необходимости восстановления ошибочно удалённых данных. Кому-то достаточно только просматривать информацию, в обязанности другого входит создание новых записей, а третьему требуется иногда вносить изменения в существующие записи.

Первая задача решается идентификацией всех пользователей информационной системы путём назначения им персональных имён, под которыми они должны авторизовываться для получения доступа к её ресурсам. В дальнейшем любые действия конкретного пользователя могут регистрироваться в системных журналах, сопровождаясь отметкой об имени учётной записи, с использованием которой эти действия были совершены.

Вторая задача решается перечислением операций, которые пользователь имеет право выполнять в информационной системе, и привязкой их к учётной записи пользователя. Операции, не входящие в список разрешённых, пользователь выполнить не сможет, и, таким образом, будет минимизирован вред, который тот в состоянии нанести своими умышленными или неумышленными действиями.

Поскольку различных операций в информационных системах, как правило, достаточно много, то перечислять разрешённые и запрещённые для каждой учётной записи утомительно. Кроме того, в ходе установки индивидуальных прав доступа легко допустить ошибку. К счастью, на практике пользователей информационной системы можно классифицировать по группам, члены которых имеют схожие должностные обязанности, а значит, должны обладать одинаковыми наборами прав для их выполнения.

Тогда для раздачи полномочий реальным пользователям можно ввести промежуточное звено — виртуальных пользователей, по одному для каждой группы, выявленной в ходе анализа должностных обязанностей. Каждому виртуальному пользователю скрупулёзно назначаются разрешения, необходимые для его нормальной работы. После этого остаётся лишь указать для каждого реального пользователя, к какой группе он относится (какому виртуальному пользователю соответствует), для того чтобы применение детального набора прав доступа происходило автоматически.

Описанный подход к разграничению доступа применяется практически во всех существующих информационных системах. Отличаются лишь терминология и детали реализации. Рассмотрим, как производится разделение прав пользователей в системе Fossil.

3.6. Пользователи и категории

Сначала выясним, какие понятия используются для управления правами пользователей системы Fossil. О том, как осуществляется назначение прав на практике, поговорим немного позже.

В системе управления доступом Fossil имеются четыре заранее настроенных виртуальных пользователя:

- nobody (никто);
- anonymous (аноним);
- reader (читатель);
- developer (разработчик).

В общих терминах управления доступом они олицетворяют группы, т. е. заранее сформированные списки разрешений, которые можно отображать на учётные записи реальных пользователей. В системе Fossil эти виртуальные пользователи называются Category (Категории). Познакомимся с ними поближе.

Категория nobody

К категории nobody относятся пользователи, попавшие на веб-страницу репозитория, но не авторизовавшиеся в системе путём ввода имени и пароля. По умолчанию такие пользователи имеют права на загрузку себе содержимого репозитория и чтение общедоступной информации: статей в подсистеме документирования (Wiki) и запросов на доработку (Tickets).

Категория anonymous

К категории anonymous относятся пользователи, которые не просто зашли на веб-страницу репозитория, но и авторизовались в системе под именем anonymous. Как было показано ранее, пароль для авторизации под этой учётной записью открыто сообщается на веб-странице с формой авторизации, поэтому такая регистрация доступна для любого одушевлённого посетителя веб-страницы.

По сравнению с категорией nobody пользователи anonymous получают возможность глубокого погружения в репозиторий через имеющиеся на веб-страницах гиперссылки, создания и дополнения запросов на доработку, дополнения статей документации.

Основная цель введения категории anonymous в дополнение к категории nobody — защита содержимого репозитория от сетевых пауков и поисковых роботов, которые сканируют сеть Интернет в автоматическом режиме для изучения всей доступной информации.

Категория reader

К категории reader относятся зарегистрированные в системе пользователи, которым явным образом назначена эта категория. Принадлежность к этой категории открывает возможность смены пароля своей учётной записи, корректировки статей документации Wiki и запросов на доработку.

Категория developer

К категории developer относятся зарегистрированные в системе пользователи, которым явным образом назначена эта категория. Пользователи, принадлежащие к этой категории, получают право на отправку в репозиторий выполненных ими изменений в файлах проекта, просмотр контактной информации других пользователей репозитория и авторов запросов на доработку, удаления записей, созданных анонимными пользователями в системах документирования и ведения заявок на доработку (право на борьбу со спамом).

Иерархия категорий

Концепция категорий в Fossil очень похожа на концепцию групп при управлении правами в операционных системах. Но если опираться на эту аналогию, то надо сразу отметить два важнейших отличия.

Во-первых, в Fossil нельзя создавать новые категории и приходится довольствоваться теми четырьмя, что перечислены выше. Учитывая ограниченный круг задач, которые решает система Fossil (по сравнению с операционной системой), на практике этого достаточно, тем более что существует возможность модифицировать набор прав, назначенных категориям.

Во-вторых, категории Fossil существуют не как самостоятельные сущности, а уже выстроены в иерархию. Так, права категории nobody автоматически добавляются к правам категории anonymous, а права категории anonymous (уже дополненные правами категории nobody), в свою очередь, автоматически назначаются категориям reader и developer. При этом наборы прав, присущие только категориям reader и developer, не пересекаются между собой.

Сказанное означает, что если при распределении прав пользователю присваивается категория reader, то нет необходимости присваивать ему ещё и категории nobody и anonymous — соответствующие разрешения будут добавлены автоматически. С другой стороны, при назначении пользователю категории developer, скорее всего, ему следует назначить и категорию reader — в противном случае он не сможет, например, изменить пароль своей учётной записи.

Владельцы и администраторы

Как уже было сказано, при создании репозитория в нём автоматически создаётся единственный пользователь — владелец этого репозитория, который обладает по отношению к созданному репозиторию всей полнотой власти. Он может выполнять над репозиторием любые операции, а также создавать и удалять других пользователей. Помимо владельца в репозитории в качестве привилегированных пользователей могут быть созданы пользователи-администраторы, которым пользователь-владелец делегирует большую часть своих полномочий.

Может возникнуть вопрос, благодаря чему учётные записи пользователей Fossil становятся обладателями привилегий? Дело в том, что помимо рассмотренных выше категорий, которым назначаются отдельные права из общего набора прав Fossil, существуют два привилегированных набора прав, детальный состав которых не раскрывается и не может быть изменён. Это как раз и есть набор прав владельца репозитория — Setup user — и набор прав администратора — Admin.

Эти привилегированные наборы прав могут быть назначены пользователям репозитория точно так же, как им назначаются категории. Основное их отличие от категорий состоит в том, что наборы прав, входящие в их состав, не могут быть изменены. Из-за того, что список прав привилегированных наборов нельзя увидеть, непросто описать и их отличие.

Если говорить кратко, то права владельца больше прав администратора. Поэтому в репозиториях небольших проектов в большинстве случаев можно обойтись без администраторов (права владельца могут быть назначены не-

скольким пользователям). Права администратора выдаются для того, чтобы пользователь мог:

- управлять пользователями репозитория;
- модерировать системы ведения документации и заявок на доработки, очищать их от спама;
- выполнять рутинные административные процедуры, следить за состоянием репозитория.

При этом надо понимать, что некоторые вещи остаются недоступными для администраторов, не являющихся владельцами репозитория.

Подводя итог, можно сказать, что управление пользователями в Fossil состоит из двух этапов:

1) создание учётной записи пользователя;

2) передача прав созданному пользователю одним из следующих способов:

- назначение ему категорий;
- назначение ему привилегированных наборов прав;
- назначение ему индивидуальных прав из набора Fossil.

Пришло время разобраться, как это осуществляется на практике.

3.7. Управление пользователями

Управление пользователями в Fossil осуществляется через веб-интерфейс. Чтобы открыть необходимую для этого панель, надо выбрать в главном меню пункт Admin, после чего перейти по гиперссылке Users (Пользователи) (рис. 3.7).

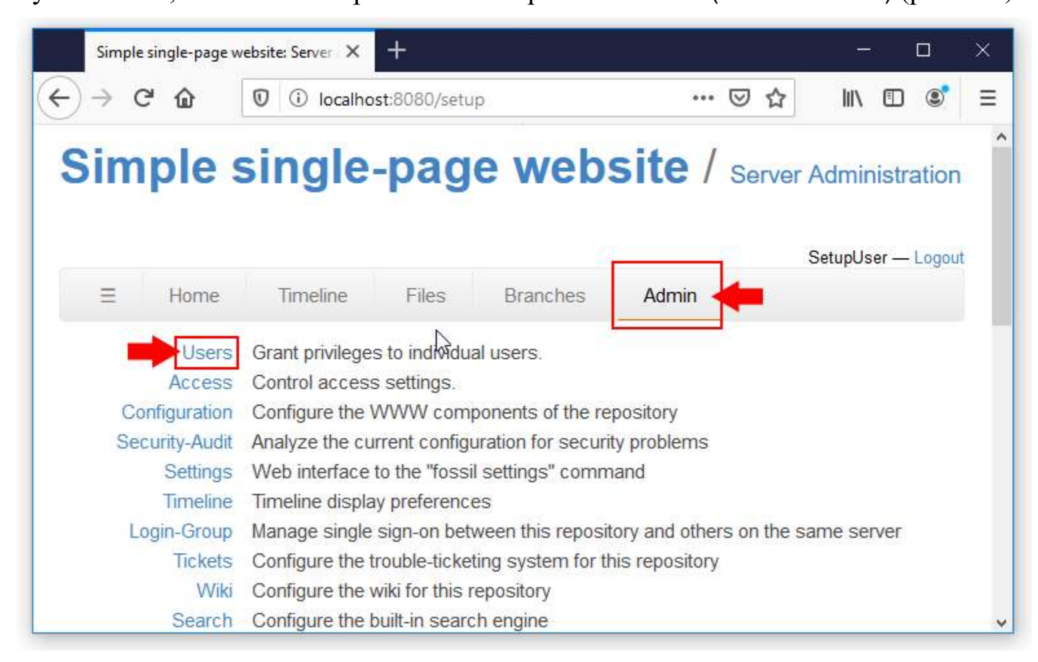


Рис. 3.7

Вызов панели управления пользователями через пункт меню Admin | Users

Внешний вид панели управления пользователями показан на рисунке 3.8. Она состоит из двух таблиц. В верхней части находится таблица категорий. Она содержит четыре строки — по одной для каждой из рассмотренных выше категорий.

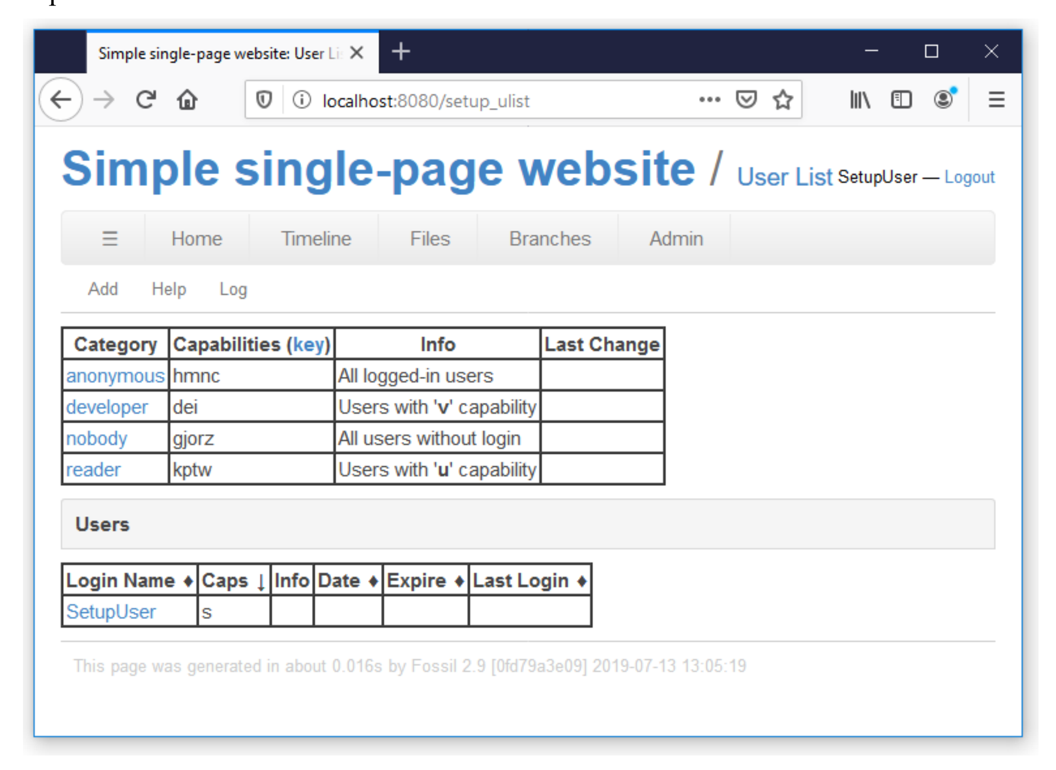


Рис. 3.8

Панель управления пользователями

В первом столбце Category (Категории) отображаются названия категорий. Они представлены гиперссылками, при переходе по которым открывается веб-страница, на которой перечислены права доступа Fossil. Возле каждого разрешения на ней имеется флажок, установка которого означает включение его в набор соответствующей категории.

Откроем, например, панель настройки категории reader (рис. 3.9). В её верхней части отображается User ID — идентификатор пользователя (для категории reader он равен 5) — и Login — имя виртуального пользователя категории (в приведенном примере рассматривается категория reader).

Центральная часть панели занята списком разрешений Capabilities, из которого отмеченные флажком включаются в набор прав для настраиваемой категории. Рядом с названиями некоторых разрешений на месте нижнего индекса записаны цветные буквы латинского алфавита. Они сигнализируют, каким категориям назначены эти разрешения: N — nobody, A — anonymous, R — reader, D — developer.

Рис. 3.9

Панель настройки категории

В поле Selected Cap (Выбранные возможности), которое находится под списком разрешений, отображается строка символов, в краткой форме представляющая выбранные для настраиваемой категории разрешения. В приведенном примере это kptw: k — Write-Wiki (написание в Вики), p — Password (изменение пароля), t — Reports (создание нового формата отчёта), w — Write-Ticket (редактирование заявки на доработку). С помощью гиперссылки key, размещённой рядом с полем Selected Cap, можно открыть справочник разрешений Fossil, в котором приведены расшифровки значений каждого символа, входящего в назначенные разрешения.

В нижней части панели настройки категории находятся кнопки Apply Changes (Применить изменения) и Cancel (Отменить). Если покинуть веб-страницу панели без нажатия кнопки Apply Changes, то внесённые изменения не будут сохранены.

Вернёмся к панели управления и продолжим рассмотрение таблицы категорий. Во втором столбце Capabilities (Возможности) отображаются строки символов, которыми кодируются права Fossil, включённые в набор прав соответствующей категории. О том, как формируются эти строки, только что было рассказано. Рядом с названием заголовка столбца находится гиперссылка key (ключ), по которой открывается справочник с полным перечнем прав.

В третьем столбце Info (Информация) отображаются описания перечисленных в таблице категорий. Из него можно узнать о том, что категория Nobody назначается всем посетителям веб-страницы проекта, категория Anonymous — всем пользователям, которые авторизовались посредством имени и пароля, а категории Reader и Developer присваиваются пользователям с помощью символов u и v соответственно.

Четвёртый столбец Last Change (Последнее изменение) должен, вероятно, заполняться датой и временем последней модификации прав соответствующей категории. Однако в версии Fossil 2.9 для Windows он остаётся пустым, из чего можно сделать вывод, что этот атрибут является зарезервированным.

В нижней части панели управления находится таблица пользователей. Новый репозиторий имеет единственного пользователя — его создателя и владельца, поэтому в таблице только одна строка.

В первом столбце Login Name (Имя входа в систему) отображается имя пользователя, которое используется на веб-странице авторизации совместно с паролем. Оно представлено гиперссылкой, при переходе по которой открывается панель настройки пользователя (рис. 3.10).

Панель настройки пользователя похожа на панель настройки категории. Поскольку учётные записи пользователей Fossil определяются своими идентификаторами User ID, имя входа в систему Login можно изменять. В новом поле Contact info (Контактная информация) можно указать дополнительные сведения о пользователе, в частности — адрес электронной почты.

Так же как и на панели настройки категорий, присутствует полный перечень разрешений Capabilities. Его возглавляют привилегированные наборы Setup и Admin, за которыми следуют категории Reader и Developer. В большинстве случаев этих пунктов достаточно для установки разрешений конкретному пользователю. Но при необходимости можно передать пользователю в индивидуальном порядке дополнительные права из основной части списка.

В поле Password (Пароль) можно изменить пароль учётной записи пользователя, который будет использоваться для входа в систему.

В отличие от категорий, список которых не может быть изменён, учётную запись пользователя можно удалить. Для этого предназначена кнопка Delete User в нижней части панели.

Вернёмся к панели управления и продолжим рассмотрение таблицы пользователей. Во втором столбце Caps (Возможности) отображается перечень прав, представленных условными символами.

В третьем столбце Info (Информация) отображаются сведения, введённые в поле Contact info панели настроек.

Рис. 3.10

Панель настройки пользователя

В чётвёртом столбце Date отображаются даты создания учётных записей пользователей. В дальнейшем они будут заменяться датами внесения последних изменений в карточки пользователей.

Пятый столбец Expire (Срок годности), исходя из его наименования, должен содержать даты, после которых учётные записи становятся недействительными. Однако в настоящее время отсутствует возможность указания этих дат через веб-интерфейс.

В шестом столбце Last Login (Последний вход в систему) будет отображаться время, прошедшее с момента последней успешной авторизации пользователя в системе.

3.8. Добавление и удаление пользователей

Для создания учётной записи нового пользователя надо воспользоваться пунктом Add (Добавить) дополнительного горизонтального меню, имеющегося на панели управления пользователями (рис. 3.11).

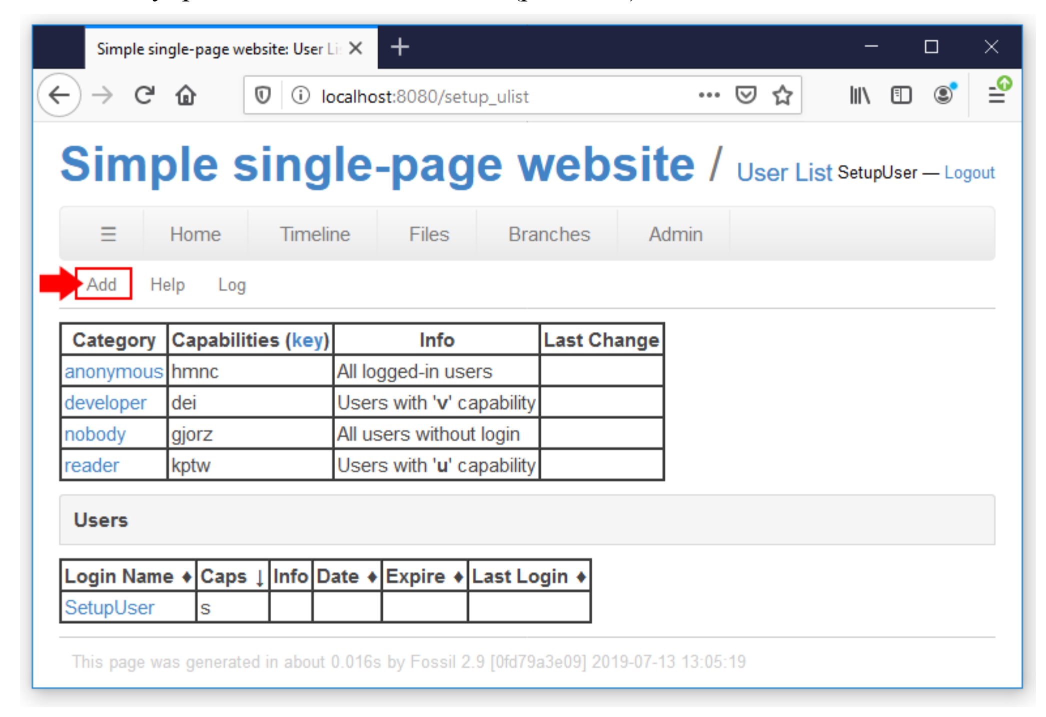


Рис. 3.11

Вызов панели добавления пользователя

Многие элементы панели добавления пользователя, которая показана на рисунке 3.12, уже знакомы по описанию панели настройки пользователя, которая была подробно рассмотрена выше. Числовой идентификатор учётной записи нового пользователя будет назначен системой автоматически. Для создаваемого пользователя достаточно указать его имя в системе (Login), контактную информацию (Contact Info), отметить предоставляемые права (Capabilities), ввести пароль (Password) и нажать кнопку Apply Changes.

В приведённой на рисунке 3.12 форме создаётся пользователь с именем Imposer1 (первый верстальщик) с адресом электронной почты imp1@fossil.server. Поскольку работа верстальщика подразумевает модификацию HTML- и CSS-файлов, составляющих исходный код веб-страниц, он должен иметь возможность отправлять результаты своей работы в репозиторий. Такие права ему предоставляет принадлежность к категориям Reader и Developer.

Рис. 3.12

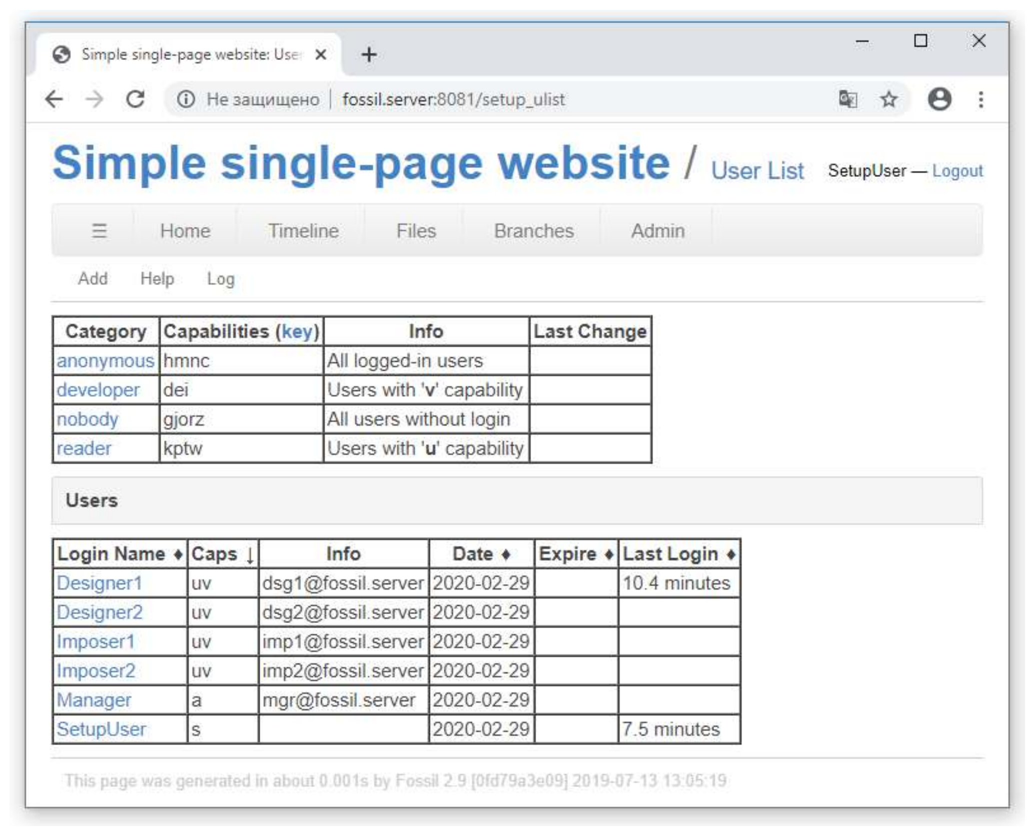
Панель добавления пользователя

Предположим, что над проектом по разработке веб-страницы будут трудиться два верстальщика и два веб-дизайнера. Дизайнеры предоставляют верстальщикам макет веб-страницы в виде изображения, а также разрабатывают визуальные элементы в виде изображений, записанных в gif- и jpeg-файлах. Верстальщики по полученному от дизайнеров макету пишут HTML- и CSS-код, который с использованием визуальных элементов обеспечит отображение разрабатываемой веб-страницы в браузере.

Создадим учётные записи верстальщиков Imposer1 и Imposer2 и дизайнеров Designer1 и Designer2 с правами на изменение содержимого репозитория.

Кроме того, создадим учётную запись для менеджера проекта Manager с правами администратора по отношению к репозиторию. Регистрационные данные (имя и начальный пароль) следует передать пользователям с рекомендацией изменить пароль на новый как можно быстрее при первом же сеансе работы в системе.

В результате проделанной работы таблица пользователей репозитория должна принять вид, показанный на рисунке 3.13. Из неё видно, что пользователь Designer1 уже воспользовался полученными регистрационными данными и десять минут назад успешно авторизовался в системе.



Category	Capabilities (key)	Info	Last Change
anonymous	hmnc	All logged-in users	
developer	dei	Users with 'v' capability	
nobody	gjorz	All users without login	
reader	kptw	Users with 'u' capability	

Login Name ♦	Caps ↓	Info	Date ♦	Expire ♦	Last Login ♦
Designer1	uv	dsg1@fossil.server	2020-02-29		10.4 minutes
Designer2	uv	dsg2@fossil.server	2020-02-29		
Imposer1	uv	imp1@fossil.server	2020-02-29		
Imposer2	uv	imp2@fossil.server	2020-02-29		
Manager	a	mgr@fossil.server	2020-02-29		
SetupUser	s		2020-02-29		7.5 minutes

Рис. 3.13

Заполненная таблица пользователей на панели управления

Может оказаться, что два дизайнера на таком небольшом проекте — непозволительная роскошь. В таком случае список пользователей придётся подкорректировать.

Во-первых, нужно удалить учётную запись второго дизайнера. Для этого надо открыть карточку удаляемого пользователя, перейдя по гиперссылке с его именем в столбце Login Name таблицы Users на странице управления учётными записями (рис. 3.14).

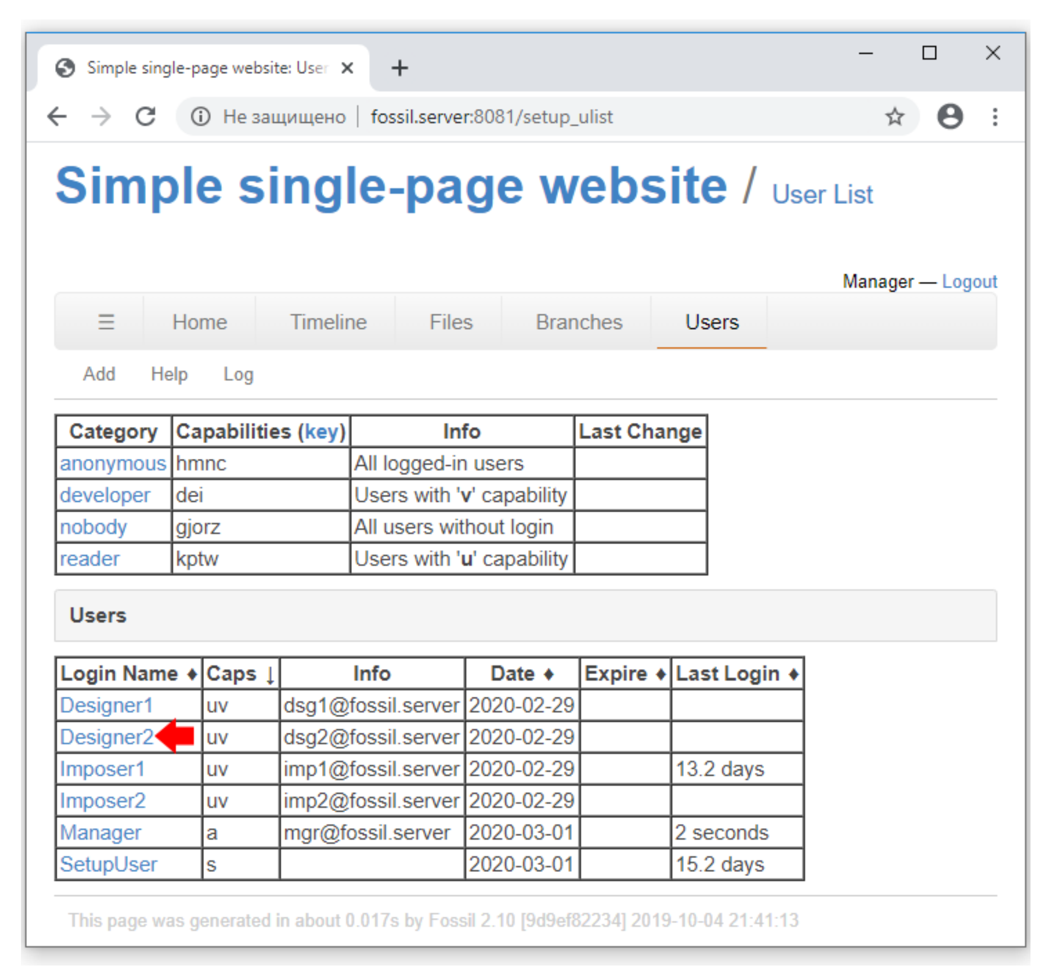


Рис. 3.14

Открытие карточки пользователя

Затем в нижней части формы с параметрами учётной записи надо нажать кнопку Delete User (Удалить пользователя). Над кнопками отобразится запрос на подтверждение удаления. Надо поставить флажок в поле Confirm delete (Подтвердить удаление) и ещё раз нажать кнопку Delete User (рис. 3.15).

Точно так же можно удалить и учётную запись первого дизайнера, а потом создать новую — для единственного дизайнера проекта. А можно подкорректировать существующую учётную запись. Для этого надо открыть карточку пользователя Designer1, как только что было описано, после чего вписать в неё имя пользователя Designer и адрес электронной почты designer@fossil.server. Будет не лишним изменить пароль учётной записи. Для сохранения внесённых изменений осталось нажать кнопку Apply Changes (Применить изменения) (рис. 3.16).

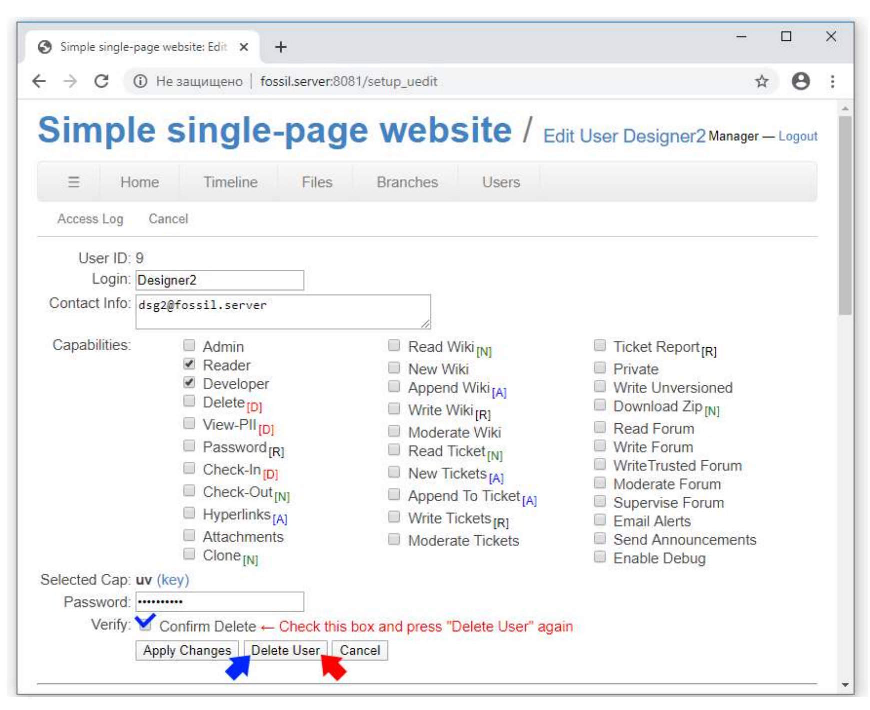


Рис. 3.15

Удаление пользователя

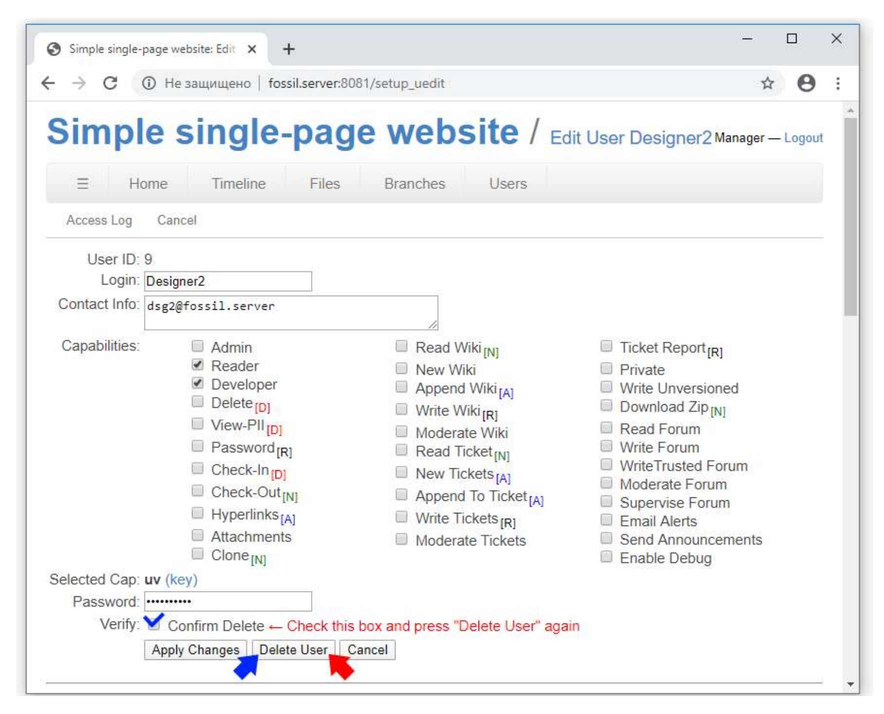


Рис. 3.16

Изменение учётной записи

В окончательном варианте команду проекта *Простой одностраничный веб-сайт* будут составлять четыре сотрудника (рис. 3.17):

- менеджер — Manager;
- первый верстальщик — Imposer1;
- второй верстальщик — Imposer2;
- дизайнер — Designer.

Users

Login Name ♦	Caps ↓	Info	Date ♦	Expire ♦	Last Login ♦
Designer	uv	designer@fossil.server	2020-02-29		
Imposer1	uv	imp1@fossil.server	2020-02-29		13.3 days
Imposer2	uv	imp2@fossil.server	2020-02-29		
Manager	a	mgr@fossil.server	2020-03-01		80.0 minutes
SetupUser	s		2020-03-01		15.2 days

Рис. 3.17

Окончательный список пользователей системы Fossil

На этом подготовительный этап можно считать завершённым. Пора приступать к реальной работе над проектом.

Глава 4
СИСТЕМА КОНТРОЛЯ ВЕРСИЙ

4.1. Начало совместной работы

4.1.1. Индивидуальные рабочие места

К настоящему моменту файл репозитория проекта «Простой одностраничный веб-сайт» создан и размещён на сервере, в репозитории заведены учётные записи участников проекта и к нему предоставлен сетевой доступ. Одним словом, всё готово для работы над проектом. Но как должен происходить рабочий процесс?

Работа над веб-сайтом представляет собой написание текстов на языках HTML и CSS и создание файлов с графическими элементами. Как правило, процесс этот итерационный и заключается в цепочке последовательных улучшений.

Это значит, что в рабочие материалы вносятся относительно небольшие изменения, решающие какую-то одну небольшую задачу, после чего оценивается получившийся результат. Если этот результат удовлетворительный, то решается очередная задача. В противном случае сделанные изменения отменяются, и осуществляется второй подход к той же задаче с учётом полученного опыта.

Изюминку в описанный процесс разработки вносит то, что над одним и тем же проектом одновременно работает несколько специалистов. И не всегда работу удаётся распределить таким образом, чтобы изменения, вносимые ими в проект, не пересекались. Если дизайнеры трудятся каждый над своим графическим элементом (например, один разрабатывает изображение логотипа, а второй — фоновое изображение), то верстальщики могут одновременно вносить изменения в одни и те же HTML-файлы.

Менеджера проекта могут интересовать рабочие материалы в текущий момент времени, чтобы контролировать прогресс, вовремя замечать возникающие затруднения и оценивать степень готовности продукта.

Система Fossil позволяет упростить взаимодействие и решить множество проблем, возникающих по ходу разработки. Но для её использования помимо серверной части необходимо настроить индивидуальные рабочие места всех участников проекта. Эта настройка заключается в выполнении двух действий:

- клонирование сетевого репозитория в локальный (требуется только при использовании сервера Fossil);
- создание рабочего каталога (требуется в том числе и при локальной разработке).

Создание индивидуального рабочего места выполняется только один раз в ходе подготовки к работе над проектом. В результате получается структура, изображённая на рисунке 4.1.

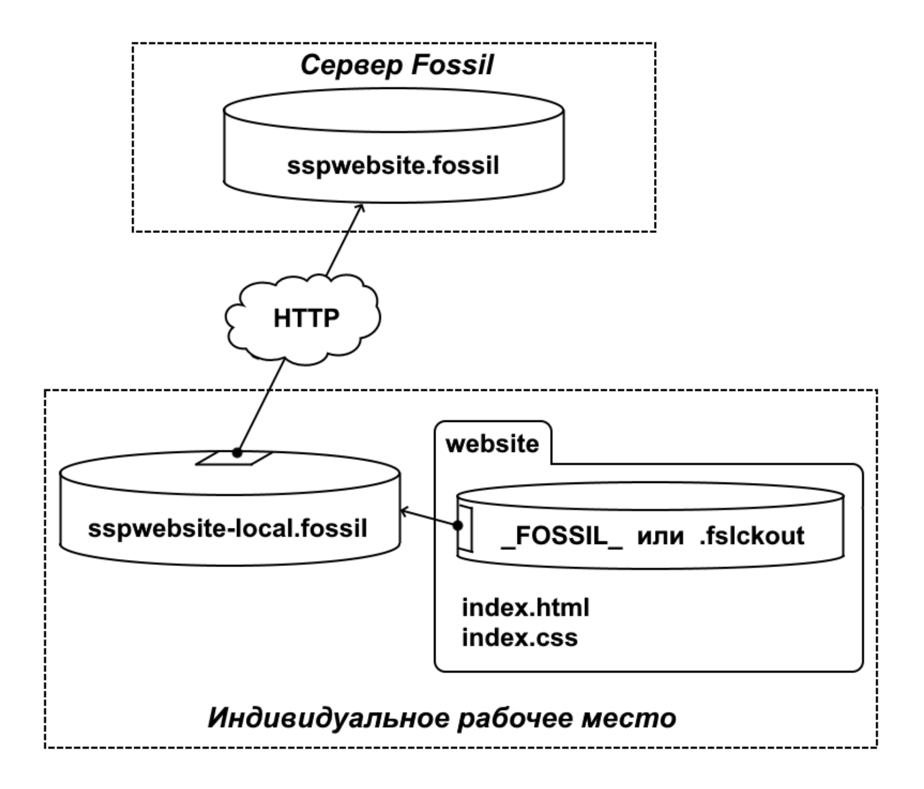


Рис. 4.1

Схема развёртывания проекта, управляемого Fossil:

цилиндрами изображены файлы, имеющие структуру базы данных SQLite. Стрелками указаны зависимости элементов.

Файл сетевого репозитория на сервере называется sspwebsite.fossil. В результате его клонирования на рабочем компьютере участника проекта создаётся файл sspwebsite-local.fossil, содержащий локальную копию сетевого репозитория. При организации рабочего каталога проекта website в нём создаётся файл _FOSSIL_ (в операционной системе Windows) или .fslckout (в других операционных системах).

Важно помнить, что локальные элементы, в отличие от серверного репозитория, содержат привязки к своим источникам: файл локального репозитория помнит, из какого сетевого репозитория он был клонирован, а рабочий каталог помнит путь к своему локальному репозиторию. Эти связи могут нарушаться при перемещении файлов, и тогда потребуется их восстановление.

Рассмотрим подробно этапы организации индивидуального рабочего места.

4.1.2. Локальный репозиторий

Первое, что необходимо сделать участникам разработки, — это получить свою копию репозитория проекта. Такая операция называется клонированием. Раньше уже говорилось о том, что репозиторий Fossil хранится целиком в одном

fossil-файле, который можно копировать на любые компьютерные системы. Однако такой вариант получения репозитория не годится для совместной работы, потому что копия файла ничего не будет знать о сервере разработки и будет считать именно себя корневым узлом.

При клонировании репозитория важно понимать, от имени какого пользователя выполняются действия и какие регистрационные данные используются. По умолчанию команда fossil использует имя учётной записи пользователя операционной системы. Это не тот пользователь, который был зарегистрирован в сетевом репозитории в качестве участника работы над проектом.

Поскольку в ходе клонирования устанавливаются доверительные отношения между сетевым и локальным репозиториями, которые будут в дальнейшем многократно использоваться, лучше на этом этапе проявить максимальную аккуратность и педантичность в указании регистрационных данных, чем потом кропотливо разбираться с причинами, по которым репозитории не синхронизируются.

Правильный подход к получению копии репозитория проекта состоит в выполнении команды:

```
fossil clone http://Imposer1@server.fossil:8081/ sspwebsite-local.fossil
```

В команде клонирования наряду с адресом сервера указывается имя пользователя для HTTP-авторизации, поэтому адрес запроса к серверу Fossil выглядит так: http://Imposer1@server.fossil:8081/, где Imposer1 — имя учётной записи участника проекта. После адреса сетевого репозитория указывается имя файла для хранения локальной копии репозитория — sspwebsite-local.fossil. Если всё было сделано правильно, то программа клонирования сначала запросит пароль указанного в адресной строке пользователя:

```
password for Imposer1:
```

После того, как пароль будет введён, появится запрос подтверждения на сохранение пароля для доступа к сетевому репозиторию:

```
remember password (Y/n)?
```

Если отказаться от сохранения пароля (для этого с клавиатуры надо ввести N), то в последующем придётся набирать его для выполнения любых действий, затрагивающих сетевой репозиторий. Причём после каждого такого действия будет предлагаться пароль сохранить. Поэтому, если нет каких-то особенных причин для отказа, его лучше сохранить, для чего ввести с клавиатуры Y либо просто нажать клавишу Enter (сохранение пароля подразумевается по умолчанию).

Клонирование репозитория сопровождается выводом примерно такого информационного блока:

```
Round-trips: 2 Artifacts sent: 0 received: 7
*** time skew *** server is slow by 19.2 minutes
Clone done, sent: 570 received: 1409 ip: 192.168.56.101
Rebuilding repository meta-data...
100.0% complete...
```

```
Extra delta compression...
Vacuuming the database...
project-id: 53e4e1445fb2a7289eb1a337ce1cf8c9a99f7b86
server-id: 8df995271ed3f0cbf7191e86605b8636faf2d2cd
admin-user: Imposer1 (password is "fcea64")
```

Он напоминает сообщение, которое было получено при создании fossil-файла с новым репозиторием. Как и в том случае, в последней строке сообщается имя создателя — владельца репозитория (Imposer1) и его пароль (fcea64). Для локального репозитория эти сведения не являются важными, поскольку доступ к панели управления локальным репозиторием при необходимости можно получить с помощью команды fossil ui без авторизации.

Во второй строке «*** time skew *** server is slow by 19.2 minutes» сообщается о том, что часы сервера отстают на 19,2 минуты от часов рабочей станции. Это плохой признак, поскольку при контроле изменений отметки времени являются важным фактором сравнения файлов. Чтобы в работе системы контроля версий не возникало нарушений, часы на компьютерах всех участников проекта должны быть синхронизированы. В современных компьютерных сетях поддержание точного времени на часах обычно не представляет трудности.

Если регистрационные данные пользователя сетевого репозитория (его имя или пароль) были указаны неправильно, то в информационном блоке будет выведена строка: Error: login failed (Ошибка: не удалось авторизоваться), а последней строкой будет сообщение: server returned an error - clone aborted (сервер вернул ошибку — клонирование прервано)».

Может возникнуть вопрос: что произойдёт, если пароль быль введён с ошибкой, но при этом было подтверждено его сохранение? Как будет происходить дальнейшая работа с репозиторием? На самом деле, ничего непоправимого не случится. Пароль запоминается в локальном репозитории, а при неправильном вводе пароля авторизация в сетевом репозитории завершится неудачей, и локальный репозиторий просто не будет создан. Чтобы решить проблему, надо просто выполнить команду ещё раз с правильным вводом пароля.

Если файл локального репозитория уже существует, то выполнение команды не приведёт к его перезаписи. Вместо этого будет выведено сообщение об ошибке:

```
file already exists: sspwebsite-local.fossil
```

Ошибкой завершится и команда, в которой указан неправильный путь к сетевому репозиторию (например, вместо номера порта 8081 ошибочно указан порт 8080):

```
cannot connect to host fossil.server:8080
Clone done, sent: 0 received: 0 ip:
server returned an error - clone aborted
```

Такое же сообщение будет выдано в случае, если попытка доступа к серверу Fossil пресекается брандмауэром.

Файл, полученный в результате клонирования сетевого репозитория, не является точной копией fossil-файла, размещённого на сервере. Во-первых, он содержит привязку к родительскому репозиторию, что позволит в дальнейшем выполнять синхронизацию рабочих материалов. Во-вторых, при клонировании не копируется секретная часть исходного репозитория, например пароли пользователей. В силу второй особенности клонирование сетевых репозиториев не считается действием, требующим особенных полномочий, и доступно даже для анонимных пользователей, желающих познакомиться с проектом.

Отметим, что пользователь Imposer1 — владелец локального репозитория, несмотря на то, что получил при клонировании такое же имя, как и пользователь сетевого репозитория — участник проекта, является совершенно самостоятельной сущностью. Он не имеет никакой связи с сетевым репозиторием.

4.1.3. Смена пароля

При эксплуатации автоматизированных компьютерных систем обычной процедурой является периодическая смена паролей пользователей. Замену паролей может инициировать администратор системы в рамках реализации политики информационной безопасности, либо пользователь может сам пожелать сменить пароль, если у него есть подозрение в его компрометации.

Изменить свой пароль пользователь может через веб-интерфейс Fossil. В приведенном примере для этого потребуется запустить веб-браузер и набрать в адресной строке URL: http://fossil.server:8081/. После авторизации надо перейти на страницу выхода из системы по гиперссылке logout, находящейся в правом верхнем углу веб-страницы (рис. 4.2).

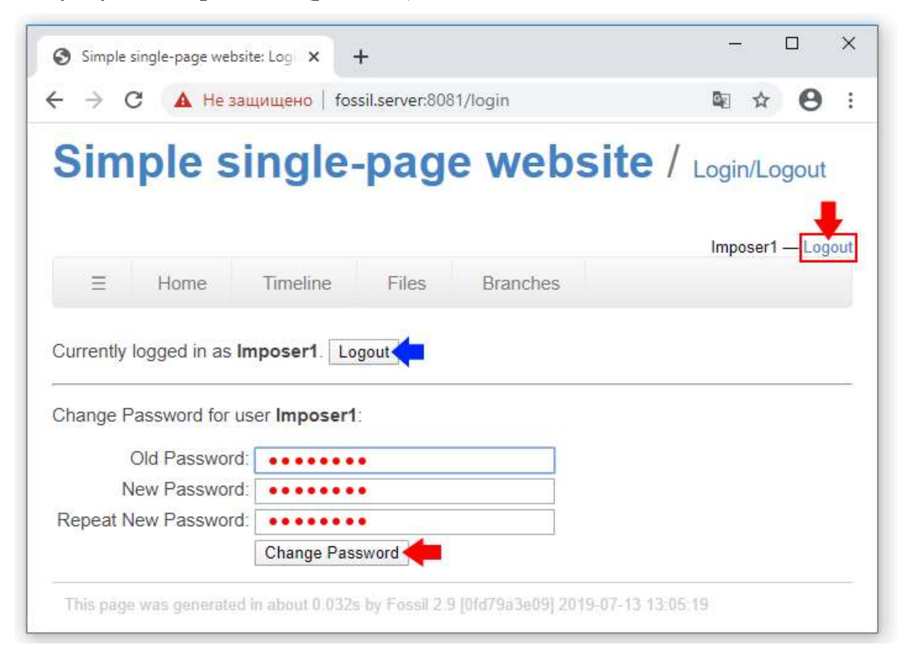


Рис. 4.2

Смена пароля пользователя сетевого репозитория

На открывшейся веб-странице наряду с кнопкой Logout, находящейся в центре и предназначенной для выхода из системы, присутствует форма смены пароля. В поле Old password надо набрать действующий пароль пользователя, в поле New password — новый пароль, а в поле Repeat New Password — повторить ввод нового пароля. Затем надо нажать кнопку Change Password.

Если действующий пароль в первом поле был набран с ошибкой, то в центре веб-страницы отобразится красный текст: You entered an incorrect old password while attempting to change your password. Your password is unchanged (Вы ввели неправильный старый пароль при попытке изменения вашего пароля. Ваш пароль не был изменён), и смены пароля не произойдёт.

Если введённые в нижних двух полях пароли не совпадают, то в центре веб-страницы отобразится красный текст: The two copies of your new passwords do not match. Your password is unchanged (Две копии нового пароля не совпадают. Ваш пароль не был изменён), и смены пароля не произойдёт.

В случае успешной смены пароля произойдёт переход на домашнюю веб-страницу проекта. И ситуация в этом случае будет выглядеть следующим образом: учётная запись на сервере имеет новый пароль, а в регистрационных данных локального репозитория содержится старый пароль. Из-за этого информационный обмен между локальным репозиторием и сервером станет невозможным.

Чтобы обновить пароль учётной записи в локальном репозитории, потребуется выполнить команду:

```
fossil remote http://Imposer1@fossil.server:8081
```

Эта команда предназначена для изменения URL, по которому доступен сетевой репозиторий, и может оказаться полезной при переносе сетевого репозитория на другой сервер. Но даже если выполнить её с тем же URL, который был указан при клонировании репозитория, она приведёт к запросу и изменению пароля учётной записи в локальном репозитории. После ввода пароля снова будет задан вопрос о его сохранении для дальнейшего использования, а в конце будет выведен установленный URL:

```
password for Imposer1: *** remember password (Y/n)? y
http://Imposer1@fossil.server:8081
```

Проверить взаимодействие локального и сетевого репозитория проще всего с помощью команды их синхронизации:

```
fossil sync -R sspwebsite-local.fossil
```

Посредством параметра *-R* указывается имя файла локального репозитория, для которого должна быть выполнена синхронизация с его сетевым источником. При отсутствии ошибок на экран будут выведены строки:

```
Sync with http://Imposer1@fossil.server:8081
Round-trips: 1 Artifacts sent: 0 received: 0
Sync done, sent: 539 received: 415 ip: 192.168.56.101
```

Если же пароли не совпадают, то в выводе команды появится строка «Error: login failed» (Ошибка: авторизация неудачна):

```
Sync with http://Imposer1@fossil.server:8081
Round-trips: 1 Artifacts sent: 0 received: 0
Error: login failed
Round-trips: 1 Artifacts sent: 0 received: 0
Sync done, sent: 539 received: 300 ip: 192.168.56.101
```

4.1.4. Рабочий каталог

Получение локальной копии сетевого репозитория — это первый этап организации индивидуального рабочего места. Работа над проектом подразумевает создание и изменение составляющих его файлов, для чего могут использоваться различные инструменты: графические и текстовые редакторы, интегрированные среды разработки, системы автоматизированного проектирования и т. п. В подавляющем большинстве все эти инструменты не умеют работать с fossil-файлами напрямую, а значит, не могут увидеть в них содержимое репозитория.

Поэтому следующим шагом в организации рабочего пространства является извлечение файлов проекта в рабочий каталог файловой системы на компьютере участника проекта. Но простая распаковка, как, например, в случае ZIP-архива, здесь тоже не годится, потому что в дальнейшем между репозиторием и рабочим каталогом будет регулярно происходить двухсторонний информационный обмен. Для упрощения этого взаимодействия необходимо сохранить связь между репозиторием и его рабочим каталогом.

Чтобы извлечь содержимое репозитория в рабочий каталог, необходимо:

1) создать рабочий каталог;

2) сделать рабочий каталог текущим;

3) открыть репозиторий из рабочего каталога.

Перечисленные шаги реализуются следующей последовательностью команд операционной системы:

```
mkdir website
cd website
fossil open ..\sspwebsite-local.fossil
```

Если команды набраны верно, то в результате выполнения последней будет выведено сообщение:

```
project-name: Simple single-page website
repository: (...)/fossil/website/..\sspwebsite-local.fossil
local-root: (...)/fossil/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
project-code: 53e4e1445fb2a7289eb1a337ce1cf8c9a99f7b86
checkout: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29
07:24:50 UTC
tags: trunk
comment: initial empty check-in (user: SetupUser)
check-ins: 1
```

В первой строке сообщается название проекта, во второй — путь к файлу репозитория, который использован в качестве источника файлов проекта, в третьей — путь к рабочему каталогу. В контексте файловых систем следует помнить о том, что в операционных системах семейства Unix в качестве разделителя элементов пути к файлу используется символ слеш /, а в операционных системах семейства Windows — обратный слеш \.

После открытия репозитория в рабочем каталоге проекта появится файл с именем _FOSSIL_ (в операционной системе Windows) или .fslckout (в других операционных системах). Он, как и файл fossil-репозитория, имеет структуру базы данных SQLite. В нём содержатся сведения о состоянии файлов проекта в рабочем каталоге и привязка к файлу локального fossil-репозитория.

4.1.5. Загрузка файла в репозиторий

На текущий момент у нас имеется сетевой репозиторий, размещённый на сервере. Один из участников проекта — верстальщик с именем Imposer1 учётной записи в системе Fossil — уже получил локальную копию сетевого репозитория и создал рабочий каталог для файлов проекта. Действия по настройке индивидуальных рабочих мест следует произвести и остальным участникам проекта — второму верстальщику Imposer2, дизайнеру Designer, а также менеджеру Manager.

А пока они это делают, первый верстальщик, дабы не терять зря времени, решил приступить к работе над проектом. В своём рабочем каталоге он создал файл index.html, содержащий стандартный скелет веб-страницы:

Листинг index.html

```
<html>
  <head>
    <title>Simple single-page website</title>
  </head>
  <body>
    <h1>Простой одностраничный веб-сайт</h1>
  </body>
</html>
```

Как теперь поделиться своей наработкой с коллегами? Надо загрузить созданный файл в репозиторий, чтобы его оттуда могли получить все желающие. Но рабочий процесс в Fossil построен так, что требуется явным образом указать, какие файлы из рабочего каталога являются частью проекта. Сначала можно выполнить команду, которая покажет, какие файлы ещё не включены в проект:

```
fossil extras
```

Для приведенного примера будет выведено имя единственного файла: index.html.

При выполнении команды обязательно должен быть текущим рабочий каталог проекта, содержащий файл _FOSSIL_ (или .fslckout). Если это не так, то на экран будет выведено сообщение об ошибке: current directory is not within an open checkout (текущий каталог не внутри открытого проекта).

Добавим файл index.html в проект:

```
fossil add index.html
```

Если всё было сделано верно, то на экране появится сообщение: ADDED index.html (добавлен index.html). Если же в имени добавляемого файла была допущена ошибка, то появится сообщение: not found: .../website/indix.html (не найден: .../website/indix.html).

Выполненная команда никак не изменила состояние ни локального, ни сетевого репозитория. Она просто включила файл index.html в список тех файлов, за которыми должна наблюдать система контроля версий. Теперь можно посмотреть состояние рабочего каталога:

```
fossil status
```

Команда выведет на экран примерно такой текст:

```
repository: …/website/..\sspwebsite-local.fossil
local-root: …/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29
07:24:50 UTC
tags: trunk
comment: initial empty check-in (user: SetupUser)
ADDED index.html
```

В последней строке сообщается о том, что система Fossil приняла на контроль файл index.html. Эту же информацию можно получить в более компактном виде, без строк заголовка, с помощью команды:

```
fossil changes
```

Наступил ответственный момент. Загрузим файл index.html в репозиторий, для чего выполним команду:

```
fossil commit
```

Начнётся процесс записи изменений в репозиторий, сопровождающийся выводом информационных сообщений:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 431 received: 338 ip: 192.168.56.101
```

На экране монитора появится окно текстового редактора с предложением ввести описание изменений, которые в настоящий момент вносятся в проект (рис. 4.3).

После ввода описания изменений, которое должно быть кратким, но содержательным (в приведенном примере это текст «Добавлен HTML-шаблон index.html»), надо выполнить сохранение в текстовом редакторе (в Блокноте достаточно нажать Ctrl + S) и завершить его работу (в Windows — с помощью комбинации клавиш Alt + F4).

```
D:\fossil\website>fossil changes
ADDED      index.html

D:\fossil\website>fossil commit
Autosync:  http://Imposer1@fossil.server:8081/
Round-trips: 1   Artifacts sent: 0  received: 0
Pull done, sent: 428  received: 338  ip: 192.168.56.101
C:\WINDOWS\notepad.exe "./ci-comment-9C78184F7AA8.txt"
```

```
ci-comment-9C78184F7AA8 – Блокнот
Файл  Правка  Формат  Вид  Справка
Добавлен HTML-шаблон index.html
# Enter a commit message for this check-in.
Lines beginning with # are ignored.
#
# user: Imposer1
# tags: trunk
#
# ADDED      index.html
```

Рис. 4.3

При загрузке изменений в репозиторий предлагается ввести описание изменений

В операционной системе Windows может быть выведено предупреждение о том, что добавляемый в репозиторий файл содержит двухсимвольные переносы строк CR/LF, за которым последует запрос подтверждения на сохранение его в таком виде:

```
./index.html contains CR/LF line endings. Use --no-warnings or the
"crlf-glob" setting to disable this warning.
Commit anyhow (a=all/c=convert/y/N)? y
```

В ответ на этот запрос надо ввести символ «y» с клавиатуры для разрешения использования такого формата текстовых файлов. Процесс загрузки файла в репозиторий будет продолжен с выводом на экран информационных сообщений:

```
New_Version:
a8ebd7176c098f320a19a1c63e7b85c8adeb80fea6cb2a535e059e4821f68dca
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 2 received: 0
Sync done, sent: 810 received: 416 ip: 192.168.56.101
```

Если теперь посмотреть состояние рабочего каталога с помощью команды fossil status, то в последней строке будет выведено описание последнего выполненного изменения с указанием пользователя, который его произвёл:

```
repository: …/fossil/website/..\sspwebsite-local.fossil
local-root: …/fossil/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02
07:13:30 UTC
```

```
parent: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29
07:24:50 UTC
tags: trunk
comment: Добавлен HTML-шаблон index.html (user: Imposer1)
```

Всё выглядит так, как будто файл index.html попал в репозиторий. Но действительно ли он уже присутствует на сервере? Для получения ответа на этот вопрос можно в адресной строке веб-браузера набрать URL сетевого репозитория: http://fossil.server:8081/ и на открывшейся веб-странице выбрать пункт горизонтального меню Timeline (Шкала времени). Отобразится веб-страница, показанная на рисунке 4.4.

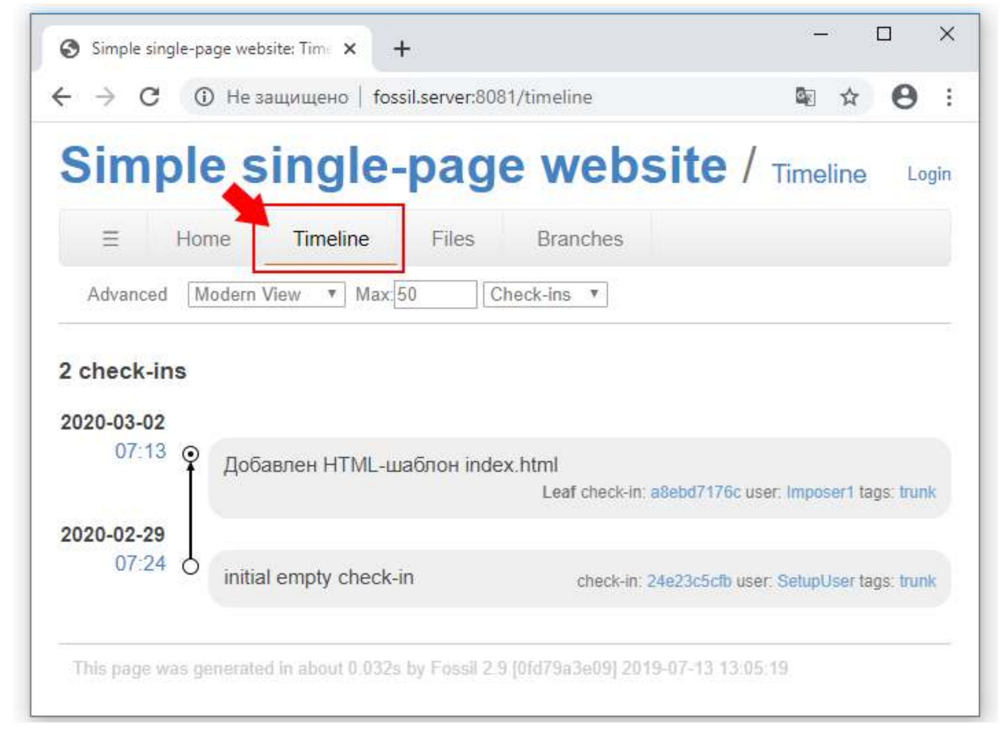


Рис. 4.4

Шкала времени репозитория проекта

На схеме видны два события, которые произошли с репозиторием с начала его существования. Первое событие, которое отображается внизу, — initial empty check-in (начальная пустая загрузка) — фиктивная отметка, являющаяся началом отсчёта времени жизни проекта. Второе событие соответствует только что выполненному добавлению файла index.html, о чём свидетельствует его описание.

4.1.6. Выгрузка обновлений из репозитория

Оставим пока что первого верстальщика Imposer1, который уже подготовил своё рабочее место, и поможем справиться с этой задачей менеджеру проекта Manager, который как раз вернулся с совещания и готов уделить производственной рутине несколько минут.

Первым делом надо получить копию сетевого репозитория:

```
fossil clone http://Manager@fossil.server:8081/ sspwebsite-local.fossil
```

Затем — создать рабочий каталог и развернуть в него репозиторий:

```
mkdir website
cd website
fossil open ..\sspwebsite-local.fossil
```

Если теперь посмотреть содержимое рабочего каталога (с помощью команды dir в Windows или команды ls в Unix/Linux), то будут отображены два файла: _FOSSIL_ или .fslckout (который, как мы помним, содержит служебную информацию о рабочем каталоге и привязку к fossil-файлу локального репозитория) и index.html.

Произошло маленькое чудо: результат работы одного сотрудника через сервер попал к другому. Менеджер проекта вполне удовлетворился таким результатом и занялся составлением графиков.

Посмотрим, как идут дела у второго верстальщика. Он уже получил локальную копию сетевого репозитория, развернул её содержимое в рабочий каталог и сейчас был занят тем, что приводил текст в файле index.html к стандарту HTML5. В результате его усилий содержимое файла index.html приобрело вид:

Листинг index.html

```
<!doctype html>
<html lang="ru">
  <head>
    <title>Simple single-page website</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Простой одностраничный веб-сайт</h1>
  </body>
</html>
```

Посмотрим, каково состояние проекта с точки зрения Fossil, для чего выполним команду:

```
fossil changes
```

Ответ уместился в одной строке:

```
EDITED index.html
```

Это означает, что система контроля версий обнаружила отличия в файле index.html, находящемся в рабочем каталоге, по сравнению с состоянием, которое хранится в локальном репозитории. Верстальщик Imposer2 готов поделиться своим видением HTML-шаблона с коллегами, однако перед этим он решил сконфигурировать репозиторий таким образом, чтобы не выдавалось предупреждение о двухсимвольных признаках завершения строки CR/LF:

```
fossil settings crlf-glob '*'
```

После этого он выполняет загрузку состояния рабочего каталога в локальный репозиторий с помощью команды fossil commit, при этом в командной

строке с помощью опции *-m* сразу указывает описание внесённых им изменений:

```
fossil commit -m "index.html приведен к стандарту HTML5"
```

Но что-то пошло не так, как ожидалось. В терминале вывелась строка: Autosync: http://Imposer2@fossil.server:8081/, и программа задумалась на несколько десятков секунд. Потом появились строки:

`cannot connect to host fossil.server:8081` (невозможно подключиться к серверу)

`Pull done, sent: 0 received: 0 ip:`

`Autosync failed.` (Автоматическая синхронизация не удалась.)

и запрос:

`continue in spite of sync failure (y/N)?` (продолжить, несмотря на ошибку синхронизации?)

Верстальщик ввёл Y для того, чтобы состояние рабочего проекта загрузилось в локальный репозиторий без синхронизации, а решение проблем в работе сети решил отложить на потом. Выполнение команды продолжилось:

```
New_Version:
d2bf34850ef82c40430e31183664efd0501e348d150ca0484ea6bd15d3f7df83
Autosync: http://Imposer2@fossil.server:8081/
cannot connect to host fossil.server:8081
Sync done, sent: 0 received: 0 ip:
Autosync failed.
```

Последняя строка говорит о том, что автоматическая синхронизация не удалась. Посмотрим, в каком состоянии находится проект с помощью команды fossil status. Вот что мы видим:

```
repository: …/sspwebsite/..\sspwebsite-local.fossil
local-root: …/sspwebsite/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02
19:01:19 UTC
parent: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02
07:13:30 UTC
tags: trunk
comment: index.html приведён к стандарту HTML5 (user: Imposer2)
```

В нижней строке записан комментарий к последней выгрузке. Значит, она уже попала в локальный репозиторий.

Через некоторое время работа компьютерной сети была восстановлена, и вестальщик Imposer2 зашёл на веб-страницу временной шкалы проекта по адресу: http://fossil.server:8081/timeline. Там последней отметкой по-прежнему значилась запись «Добавлен HTML-шаблон index.html» от пользователя Imposer1. Значит, в удалённый репозиторий последние изменения пока что не попали.

Чтобы синхронизировать репозитории, верстальщик Imposer2 ввёл команду:

```
fossil sync
```

В трёх строках, которые были выведены на экран в результате выполнения команды, нет упоминаний о каких-либо проблемах:

```
Sync with http://Imposer2@fossil.server:8081/
Round-trips: 2 Artifacts sent: 2 received: 0
Sync done, sent: 1478 received: 979 ip: 192.168.56.101
```

А для проверки состояния сетевого репозитория потребовалось лишь обновить окно веб-браузера (рис. 4.5).

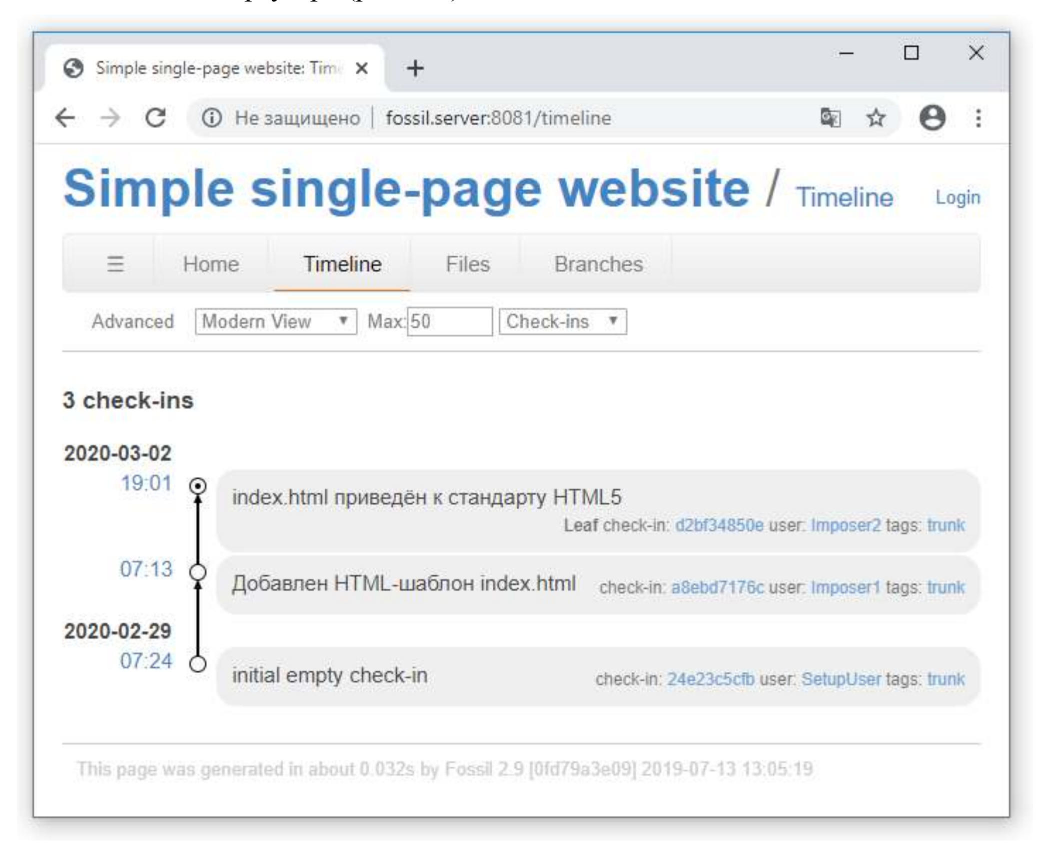


Рис. 4.5

Шкала времени на сетевом репозитории после внесения изменений в файл index.html

Тем временем первый верстальщик решил проверить, не произошло ли каких-нибудь изменений в сетевом репозитории и ввёл команду fossil sync. В полученных сообщениях не было предупреждений об ошибках:

```
Sync with http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Sync done, sent: 615 received: 488 ip: 192.168.56.101
```

Поэтому он решил посмотреть обновлённое состояние своего проекта с помощью команды fossil status. Полученный ответ показал отсутствие изменений, в последней строке по-прежнему значился его комментарий:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02
07:13:30 UTC
parent: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29
07:24:50 UTC
child: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02
19:01:19 UTC
tags: trunk
comment: Добавлен HTML-шаблон index.html (user: Imposer1)
```

Удивлённый, что до сих пор ничего не произошло, он позвонил второму верстальщику и был крайне озадачен его утверждением, что тот давно поправил файл index.html и что сделанные им исправления отражены в сетевом репозитории.

В поисках слабого звена верстальщик Imposer1 посмотрел шкалу времени в своём локальном репозитории, запустив локальный веб-интерфейс командой fossil ui ..\sspwebsite-local.fossil и выбрав в нём пункт меню Timeline. В верхней части шкалы отображалась запись «index.html приведён к стандарту HTML5», сделанная верстальщиком Imposer2. Это значит, что локальный репозиторий находится в актуальном состоянии относительно сетевого репозитория, синхронизация репозиториев действительно прошла успешно. Почему же изменения, внесённые вторым верстальщиком, не попали в рабочий каталог?

Побродив по страницам документации проекта Fossil, верстальщик Imposer1 понял свою ошибку. Он полагал, что команда fossil sync, выполненная из рабочего каталога, осуществляет как обмен изменениями между локальным и сетевым репозиториями, так и отражение нового состояния локального репозитория на рабочий каталог. На самом деле это не так, в чём он только что смог убедиться. Эта команда действительно синхронизирует репозитории, но рабочий каталог при этом не затрагивается.

Для отображения состояния репозитория, к которому привязан рабочий каталог, на файлы рабочего каталога, надо выполнить дополнительную команду. Первый верстальщик ввёл её с клавиатуры:

```
fossil update
```

В окне терминала отобразился результат выполнения команды:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 394 received: 484 ip: 192.168.56.101
UPDATE index.html ОБНОВЛЁН index.html
-------------------------------------------------------------------
updated-to: 2d438b323cb45a51ef612f64765439e85da5dd04 2020-03-03 08:42:46 UTC
```

```
tags: trunk
comment: index.html приведен к стандарту HTML5 (user: Imposer2)
changes: 1 file modified. изменения: 1 файл модифицирован.
"fossil undo" is available to undo changes to the working checkout.
```

Из полученных сообщений отчётливо видно, что файл index.html наконец-то обновился. В последней строке содержится подсказка, как откатить полученные изменения, если после них проект вдруг перестанет работать: надо выполнить команду fossil undo. Пока же всё идёт так, как и ожидалось с самого начала. Если теперь выполнить команду fossil status, то будет получен ответ:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 2d438b323cb45a51ef612f64765439e85da5dd04 2020-03-03 08:42:46 UTC
parent: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02 07:13:30 UTC
tags: trunk
comment: index.html приведен к стандарту HTML5 (user: Imposer2)
```

Если сравнить полученные сведения о состоянии проекта с теми, которые получил второй верстальщик, то можно увидеть, что они совпадают начиная с параметра «parent:». Наконец-то рабочий каталог у обоих верстальщиков получил одинаковое содержимое.

Сейчас можно отметить ещё одно важное наблюдение. Команда fossil status выводит информацию об изменениях, которые произошли в рабочем каталоге с момента последнего информационного обмена с репозиторием. Она не производит сравнение текущего состояния рабочего каталога с состоянием локального репозитория, как можно было бы подумать. Именно поэтому даже в то время, когда локальный репозиторий был уже синхронизирован с сетевым репозиторием командой fossil sync, команда fossil status ничего не сообщала о его новом состоянии.

4.2. Конфликт изменений

4.2.1. Одновременное редактирование файла

Воодушевившись первым успехом, оба верстальщика принялись за разработку веб-страницы. Очевидно, она не сможет обойтись без стилевого оформления.

Первый верстальщик решил разместить стили во внешнем файле, для чего создал файл index.css, содержащий пока что единственную строку:

```
* {margin: 0; padding: 0}
```

А в файл index.html в конец элемента <head> добавил строку:

```
<link href="index.css" rel="stylesheet" type="text/css">
```

Пока первый верстальщик проверял результат своей работы, второй верстальщик в конец элемента <head> добавил стилевой блок:

```
<style>
  * {margin: 0; padding: 0}
</style>
```

После внесения изменений и сохранения файла index.html он решил посмотреть, чем отличаются исправленные варианты файлов в рабочем каталоге проекта от моментального снимка, хранящегося в репозитории. Для этого второй верстальщик ввёл команду:

```
fossil diff
```

Команда вывела информацию о различиях в следующем виде:

```
Index: index.html
================================================================
--- index.html
+++ index.html
@@ -1,10 +1,13 @@
 <!doctype html>
 <html lang="ru">
   <head>
     <title>Simple single-page website</title>
     <meta charset="utf-8">
+    <style>
+      * {margin: 0; padding: 0}
+    </style>
   </head>
   <body>
     <h1>Простое одностраничное веб-приложение</h1>
   </body>
 </html>
```

В первой строке Index: сообщается, что дальнейшие сведения относятся к файлу index.html. После разделительной черты из знаков равенства = идут две строки с именами сравниваемых файлов: символами --- обозначено имя исходного файла, символами +++ обозначено имя результирующего файла. В данном случае сравнивается прошлый файл index.html с файлом нынешним, поэтому имена совпадают.

Затем идёт заголовок блока различий. Это строка, которая начинается и заканчивается символами @@, между которыми находятся две пары чисел, разделённых запятой. Пара чисел, обозначенная знаком минус, относится к исходному файлу, а знаком плюс — к результирующему.

Первое число пары означает номер строки в файле (исходном или результирующем), с которой начинается следующий после заголовка блок различий, а второе число — количество строк в файле (исходном или результирующем), замещаемых этим блоком различий. В рассматриваемом примере блок различий должен быть наложен на начало обоих файлов (строка номер 1) и соответствует всему их содержимому (10 строк в исходном и 13 строк в результирующем).

Далее, с отступом в один символ от левой границы, идёт содержимое блока различий, т. е. строки, отражающие результат сравнения. Символ, находящийся непосредственно у левой границы, означает следующее:

- пробел — строка присутствует как в исходном файле, так и в результирующем, в неизменном виде;
- знак плюс (+) — строка отсутствует в исходном файле, но присутствует в результирующем (добавлена);
- знак минус (-) — строка присутствует в исходном файле, но отсутствует в результирующем (удалена).

В общем случае таких блоков различий может быть несколько — по одному для каждого различающегося участка сравниваемых файлов. Неизменные строки в блоках различий помогают установить контекст, к которому относятся содержащиеся в блоке описания различий.

В приведенном примере, как и ожидалось, знаками + обозначены строки, добавленные в заголовок HTML-кода веб-страницы.

Удовлетворённый проделанной работой, второй верстальщик загрузил новое состояние рабочего каталога в репозиторий проекта командой:

```
fossil commit -m "добавлен стилевой блок"
```

Процедура загрузки прошла без ошибок:

```
Autosync: http://Imposer2@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 431 received: 488 ip: 192.168.56.101
New_Version:
38b8a0865f8e4642cec82705f51b4d564dac0ca21629bcd194ed6bd69e719f8d
Autosync: http://Imposer2@fossil.server:8081/
Round-trips: 1 Artifacts sent: 2 received: 0
Sync done, sent: 924 received: 560 ip: 192.168.56.101
```

Тем временем первый верстальщик занялся подготовкой загрузки своего рабочего каталога в репозиторий. Поскольку он не только вносил изменения в существующий файл проекта, но и создал новый файл, то решил посмотреть перечень файлов рабочего каталога, которые пока что не включены в контроль версий, с помощью команды:

```
fossil extras
```

Выведенный список, как и ожидал верстальщик, содержал единственный файл:

```
index.css
```

Верстальщик включил его в контроль версий с помощью команды:

```
fossil add index.css
```

На экране появилось подтверждение выполненной операции:

```
ADDED index.css
```

Теперь всё готово к загрузке проделанной работы в репозиторий.

4.2.2. Начало конфликта

Чтобы загрузить состояние рабочего каталога в репозиторий, верстальщик Imposer1 ввёл команду:

```
fossil commit –m "к index.html подключён файл с таблицей стилей index.css"
```

В ответ он получил неожиданное сообщение

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 2
Pull done, sent: 913 received: 1366 ip: 192.168.56.101
Would fork. "update" first or use --branch or --allow-fork.
```

Последняя строка сообщает: «Требуется создать ответвление. Выполните сначала команду update или используйте опции *--branch* или *--allow-fork*». Чтобы посмотреть, в каком состоянии сейчас находится его локальный проект, верстальщик Imposer1 выполнил команду:

```
fossil status
```

Ответ выглядел следующим образом:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02 19:01:19 UTC
parent: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02 07:13:30 UTC
child: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
tags: trunk
comment: index.html приведён к стандарту HTML5 (user: Imposer2)
ADDED index.css
EDITED index.html
```

Последним по-прежнему значится комментарий верстальщика Imposer2 о приведении файла index.html к стандарту HTML5, файл index.css отмечен как представленный к включению в контроль версий, а файл index.html значится модифицированным. Из всего этого следует, что попытка выгрузки завершилась неудачно, состояние репозитория не изменилось.

Чтобы разобраться с причинами неудачи, верстальщик Imposer1 посмотрел шкалу проекта в веб-браузере по ссылке http://fossil.server:8081/timeline и обнаружил свежую запись с комментарием «добавлен стилевой блок», сделанную вторым верстальщиком. Для того, чтобы увидеть детали скрывающихся за этой записью изменений, он перешёл по гиперссылке с идентификатором записи 38b8a0865f (на самом деле в веб-интерфейсе отображаются лишь первые десять шестнадцатеричных цифр идентификатора) (рис. 4.6).

Веб-страница, которая отобразилась после перехода по гиперссылке, состоит из трёх частей (рис. 4.7). В верхней части, озаглавленной Overview (Обзор), содержится общая информация об изменении: комментарий, полный идентификатор, имя пользователя-инициатора, дата и время внесения изменения.

В средней части, озаглавленной Context (Контекст), отображается элемент временной шкалы, содержащий выделенную цветом запись о текущем изменении, а также запись, непосредственно предшествующую просматриваемому изменению.

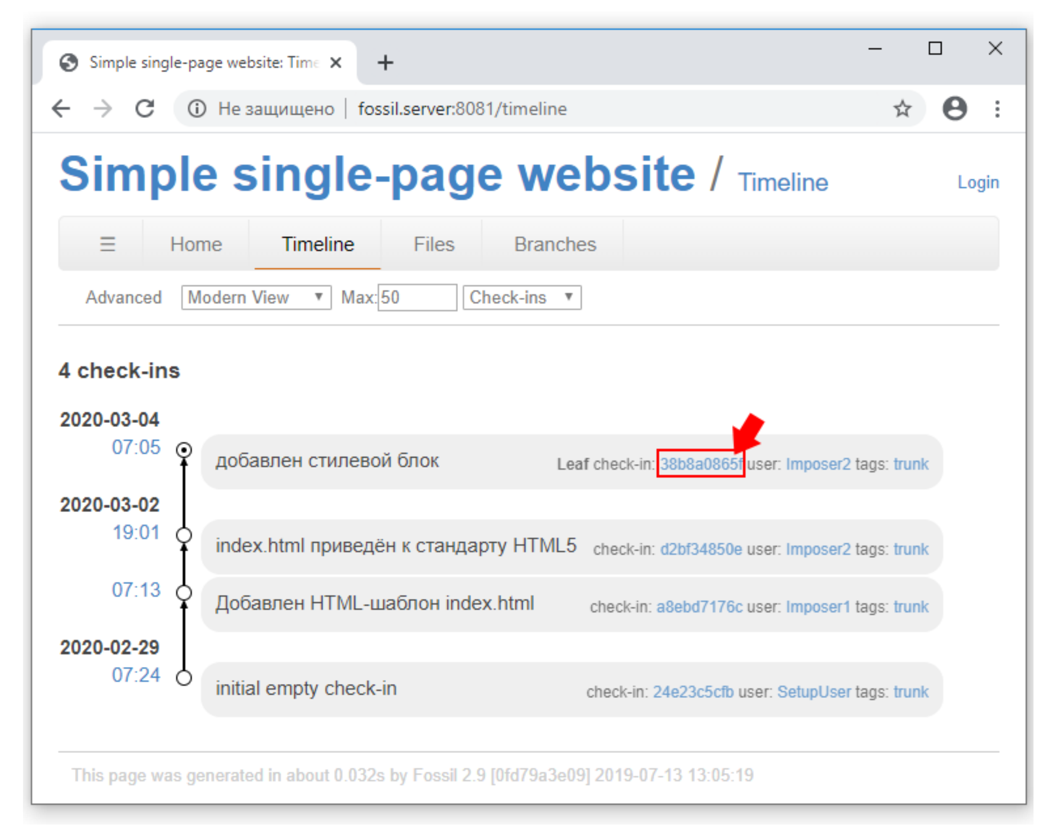


Рис. 4.6

Для просмотра деталей изменений, скрывающихся за записью на шкале времени, можно перейти по гиперссылке с идентификатором записи

Детальные сведения об изменении содержатся в нижней части, которая озаглавлена Changes. Сверху у неё имеется своё горизонтальное меню, с помощью которого можно менять способ отображения деталей изменения. По умолчанию область отображения разделена вертикальной границей на две части. В левой части показано состояние файла до изменения, а в правой части — после. При этом изменённые строки выделены цветом.

Если на экране такой вариант отображения выглядит слишком мелким, то можно переключить режим отображения на вариант Unified Diffs (Унифицированный формат) с помощью одноимённого пункта меню. После этого просмотр деталей изменения становится более комфортным (рис. 4.8).

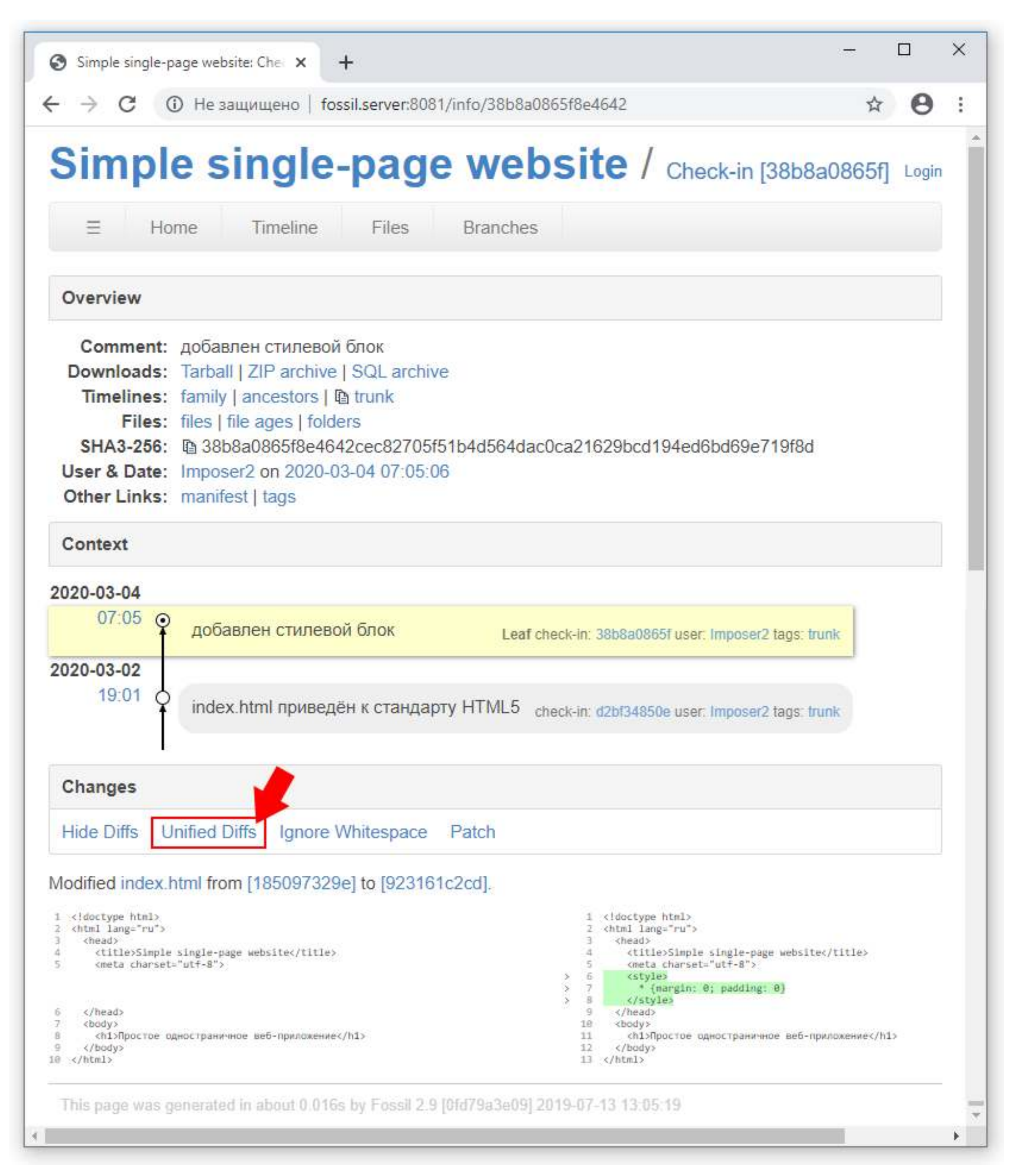


Рис. 4.7

Веб-страница с информацией об изменении

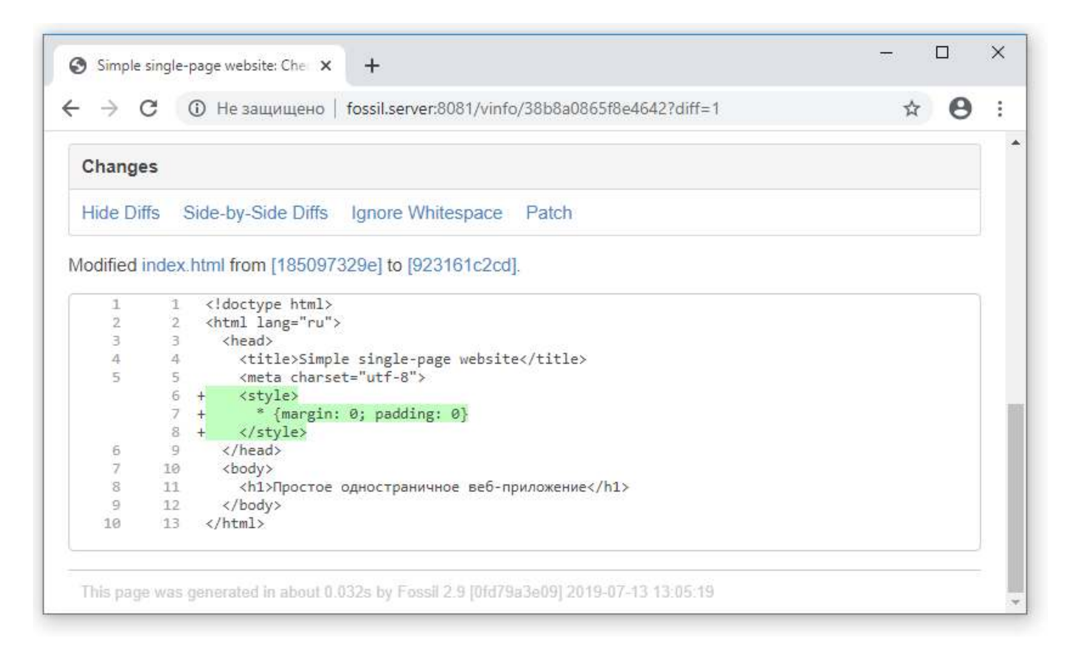


Рис. 4.8

Просмотр изменений в режиме Unified Diffs (Унифицированный формат)

Для просмотра различий между моментальным снимком рабочего каталога, загруженного верстальщиком Imposer2 в репозиторий под идентификатором 38b8a0865f, и своим рабочим каталогом верстальщик Imposer1 ввёл команду:

```
fossil diff -r 38b8a0865f
```

Команда вывела подробную информацию о различиях:

```
ADDED    index.css
Index: index.html
===============================================================
--- index.html
+++ index.html
@@ -1,13 +1,11 @@
 <!doctype html>
 <html lang="ru">
   <head>

     <title>Simple single-page website</title>
     <meta charset="utf-8">
-    <style>
-      * {margin: 0; padding: 0}
-    </style>
+    <link href="index.css" rel="stylesheet" type="text/css">
   </head>
   <body>
     <h1>Простое одностраничное веб-приложение</h1>
   </body>
 </html>
```

Теперь стало очевидно, что система контроля версий оказалась в затруднительном положении, поскольку оба верстальщика предложили два разных направления развития проекта. И, поскольку второй верстальщик успел выгрузить свои изменения раньше, искать выход из сложившейся ситуации придётся первому верстальщику.

4.2.3. Обострение конфликта

Следуя рекомендации Fossil, верстальщик Imposer1 выполнил команду на получение изменений из репозитория в свой рабочий каталог:

```
fossil update
```

Команда завершилась успешно с выдачей нескольких информационных строк:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 397 received: 560 ip: 192.168.56.101
MERGE index.html СЛИЯНИЕ index.html
***** 1 merge conflicts in index.html 1 конфликт слияния в index.html
-------------------------------------------------------------------------
updated-to: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
tags: trunk
comment: добавлен стилевой блок (user: Imposer2)
changes: 2 files modified.
WARNING: 1 merge conflicts ВНИМАНИЕ: 1 конфликт слияния
"fossil undo" is available to undo changes to the working checkout.
```

Верстальщику Imposer1 стало чрезвычайно любопытно, как поступил Fossil с противоречиями в файле index.html, и его руки потянулись к текстовому редактору. Но он сдержал этот порыв и решил разобраться с ситуацией последовательно, для чего сначала посмотрел состояние рабочего каталога командой:

```
fossil status
```

Команда вывела такое сообщение:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04
07:05:06 UTC
parent: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02
19:01:19 UTC
tags: trunk
comment: добавлен стилевой блок (user: Imposer2)
ADDED index.css ДОБАВЛЕН index.css
CONFLICT index.html КОНФЛИКТ index.html
```

Если текущее состояние сравнить с состоянием, в котором находился рабочий каталог непосредственно перед выполнением команды fossil update, то можно заметить два отличия. Во-первых, произошло переключение точки от-

счёта изменений от состояния «index.html приведён к стандарту HTML5 (user: Imposer2)» к состоянию «добавлен стилевой блок (user: Imposer2)». То есть первый верстальщик уступил второму и принял его изменения в свой локальный репозиторий.

Во-вторых, последние изменения, выполненные первым верстальщиком, пересчитались для нового состояния репозитория. Второй верстальщик не создавал файла с именем index.css, поэтому его отметка «добавлен» осталась прежней. А файл index.html, в который первый верстальщик вставил строку для привязки к нему файла с таблицами стилей index.css и в который в это же время второй верстальщик добавил стилевой блок, вместо отметки «отредактирован» получил отметку «конфликт».

Наконец верстальщик Imposer1 открыл в текстовом редакторе файл index.html, чтобы посмотреть суть конфликта. И вот что он увидел:

```
<!doctype html>
<html lang="ru">
  <head>
    <title>Simple single-page website</title>
    <meta charset="utf-8">
<<<<<<< BEGIN MERGE CONFLICT: local copy shown first <<<<<<<<
    <link href="index.css" rel="stylesheet" type="text/css">
  </head>
======= COMMON ANCESTOR content follows ================
  </head>
======= MERGED IN content follows ======================
    <style>
      * {margin: 0; padding: 0}
    </style>
  </head>
>>>>>>> END MERGE CONFLICT >>>>>>>>>>>>>>>>>>>>>>>>>>>>
  <body>
    <h1>Простое одностраничное веб-приложение</h1>
  </body>
</html>
```

Система Fossil попыталась в меру своих способностей объединить два варианта файла index.html — со стилевым блоком, вставленным верстальщиком Imposer2, и с привязкой файла с таблицами стилей, реализованной верстальщиком Imposer1 — в один файл. Строка BEGIN MERGE CONFLICT: local copy shown first (НАЧАЛО КОНФЛИКТА СЛИЯНИЯ: сначала показана локальная копия) отмечает начало комбинированного блока, а строка END MERGE CONFLICT (КОНЕЦ КОНФЛИКТА СЛИЯНИЯ) — его конец. Строкой COMMON ANCESTOR contents follows отмечена общая часть конфликтующих участков текста, а после строки MERGED IN content follows следует блок текста, полученный из сетевого репозитория.

Помимо этого, если сейчас посмотреть содержимое рабочего каталога, в нём можно обнаружить три новых файла — различные варианты index.html:

- index.html-baseline — вариант файла index.html непосредственно перед внесением в него конфликтующих изменений (как первым, так и вторым верстальщиком);
- index.html-original — вариант файла index.html верстальщика Imposer1 до того, как система Fossil произвела его модификацию из-за возникшего конфликта (своеобразная резервная копия);
- index.html-merge — вариант файла index.html, полученный из репозитория (с изменениями, выполненными верстальщиком Imposer2).

Очевидно, что структура файла index.html, находящегося в рабочем каталоге верстальщика Imposer1, сейчас нарушена. Верстальщику необходимо взять текстовый редактор и принять непростое решение относительно того, чей вариант развития проекта оставить. Вероятно, он созвонится со вторым верстальщиком, они обсудят преимущества и недостатки обоих вариантов и примут согласованное решение.

4.2.4. Уступает второй верстальщик

Предположим, что верстальщик Imposer2 согласился с тем, что лучше максимально отделить оформление веб-сайта от его содержания и вынести его в отдельный файл index.css, как это сделал верстальщик Imposer1. Тогда Imposer1 должен привести файл index.html к тому виду, который был до того, как система Fossil выразила в нём своё видение ситуации, и снова выполнить команду:

```
fossil commit -m "к index.html подключён файл с таблицей стилей index.css"
```

Так и поступил первый верстальщик. Он открыл редактор, убрал созданный вторым верстальщиком стилевой блок и внесённые системой Fossil маркерные строки, при этом оставил свою ссылку на index.css, переключился на окно командной строки и выполнил приведенную команду. Однако в ответ получил сообщение об ошибке:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 430 received: 560 ip: 192.168.56.101
possible unresolved merge conflict in ./index.html
abort due to unresolved merge conflicts; use --allow-conflict to override
```

В последних двух строках говорится следующее:

```
возможно неразрешённый конфликт слияния в ./index.html прервано по
причине неразрешённых конфликтов; используйте --allow-conflict to
override
```

Оказывается, верстальщик Imposer1 забыл сохранить изменения из текстового редактора в файл index.html. Быстро исправив свою оплошность, он решил взглянуть на состояние рабочего каталога с помощью команды fossil status. Теперь предупреждение CONFLICT (КОНФЛИКТ) у файла index.css превратилось в обычную рабочую отметку EDITED (ОТРЕДАКТИРОВАН):

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
```

```
checkout: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
parent: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02 19:01:19 UTC
tags: trunk
comment: добавлен стилевой блок (user: Imposer2)
ADDED index.css
EDITED index.html
```

Повторив приведенную выше команду fossil commit..., первый верстальщик наконец-то загрузил результат своего труда в репозиторий:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 431 received: 560 ip: 192.168.56.101
New_Version:
0ed967045e7cd7ed203edc0836347c61acde5c20c8a506ca3a3046269cc8033a
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 3 received: 0
Sync done, sent: 1107 received: 670 ip: 192.168.56.101
```

Конечно же, верстальщик Imposer1 не смог устоять перед искушением посмотреть состояние сетевого репозитория. Вот что он увидел, перейдя по ссылке http://fossil.server:8081/timeline на шкалу времени проекта (рис. 4.9).

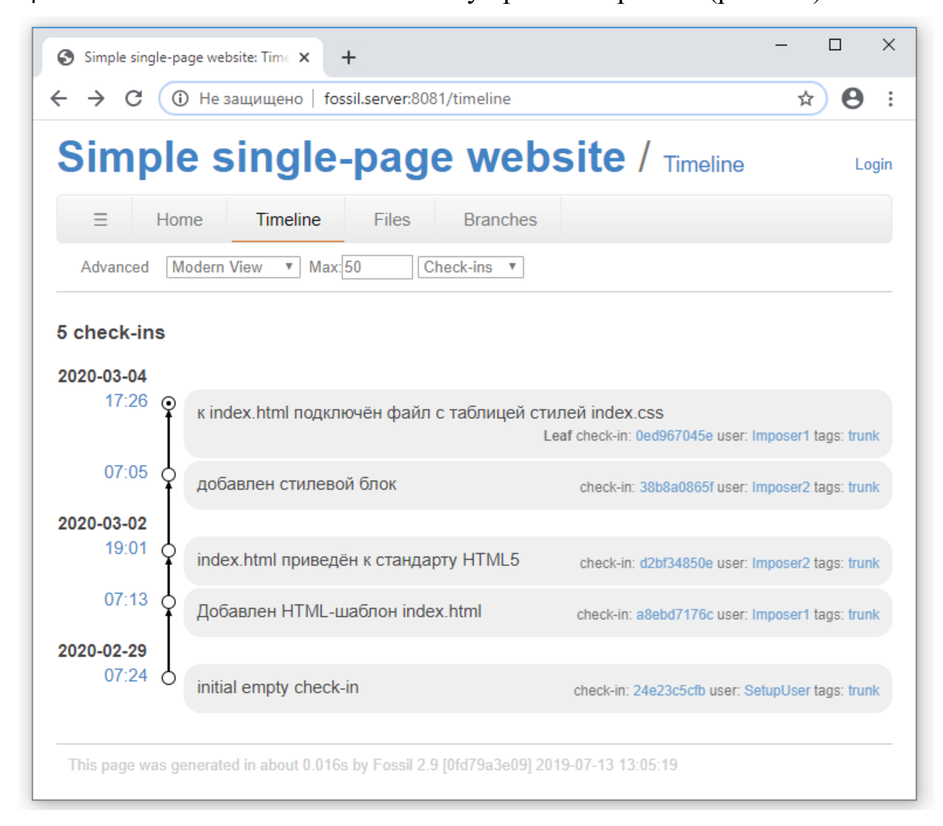


Рис. 4.9

Шкала времени проекта: победил вариант первого верстальщика

А детальные изменения в режиме Unified Diffs выглядят, как показано на рисунке 4.10. На нём видно, что в качестве исходного варианта для модификации файла index.html был взят предложенный верстальщиком Imposer2 текст, содержащий стилевой блок в строках 6–8. Последними изменениями, которые предоставил верстальщик Imposer1, эти три строки удалены (они отмечены знаками «минус» и выделены красным цветом), и их место заняла строка, содержащая ссылку на файл index.css (она отмечена знаком «плюс» и зелёным цветом).

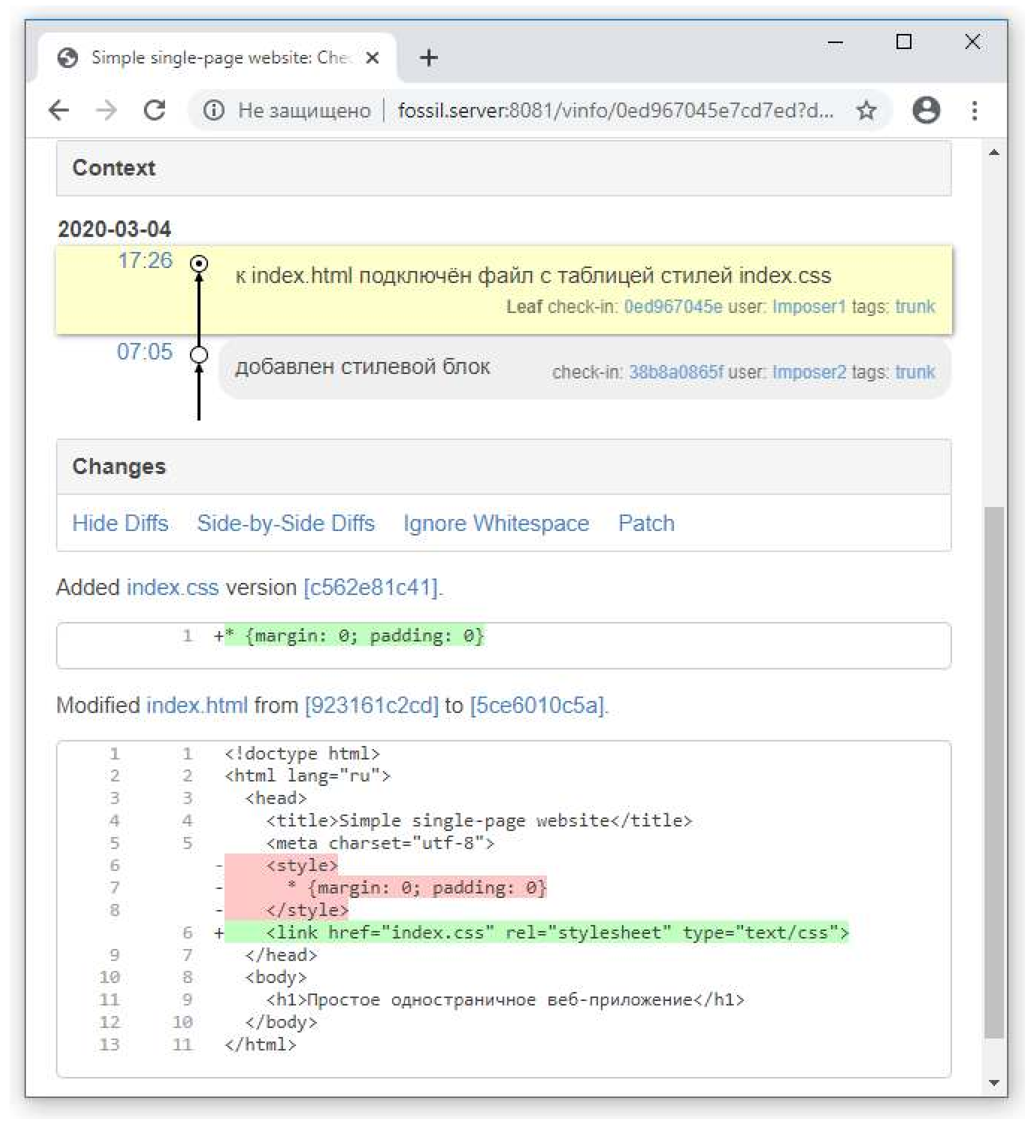


Рис. 4.10

Детальные изменения при победе варианта первого верстальщика

4.2.5. Уступает первый верстальщик

Вернёмся к моменту обсуждения наилучшего варианта подключения стилей к веб-сайту. Предположим, что верстальщики пришли к выводу, что поскольку сайт является одностраничным, то лучше и стили поместить в html-файл — тогда они гарантированно будут загружены вместе с основным содержимым. Таким образом, Imposer1 пожертвовал проделанной работой в пользу решения, предложенного Imposer2. Но как отобразить это решение на состояние репозитория?

Первый верстальщик может исключить файл index.css из системы контроля версий с помощью команды:

```
fossil rm index.css
```

Если команда выполнится удачно, на экран будет выведено:

```
DELETED index.css (УДАЛЁН index.css)
```

В этом сообщении слово «УДАЛЁН» означает удаление лишь из системы контроля версий (исключение из списка файлов, подконтрольных системе Fossil), поэтому удаление его из файловой системы нужно произвести отдельно, средствами операционной системы компьютера. В качестве окончательного варианта файла index.html можно взять файл index.html-merge, который был создан системой контроля версий и содержит текст, предоставленный вторым верстальщиком.

После выполнения описанных действий команда fossil status покажет отсутствие изменений в рабочем каталоге по отношению к последнему состоянию проекта — варианту, предложенному вторым верстальщиком:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
parent: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02 19:01:19 UTC
tags: trunk
comment: добавлен стилевой блок (user: Imposer2)
```

Теперь можно попытаться загрузить состояние рабочего каталога в репозиторий:

```
fossil commit -m "победил вариант, предложенный Imposer2"
```

Как можно было ожидать, ничего не произошло:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 435 received: 560 ip: 192.168.56.101
nothing has changed; use --allow-empty to override
```

Сообщение в последней строке говорит: «ничего не изменилось; используйте --allow-empty для переопределения».

В принципе, такой план действий для решения конфликта уместен. Но более рациональным с точки зрения верстальщика Imposer1 может быть реше-

ние о переводе своего рабочего каталога к последнему состоянию, зафиксированному в сетевом репозитории. А это как раз и будет последняя загрузка, произведенная в репозиторий верстальщиком Imposer2.

Выполнить описанную процедуру можно одной командой:

```
fossil checkout --latest --force
```

Она выведет на экран имена файлов в локальном проекте, которые будут затронуты процедурой отката. В рассматриваемом примере это единственный файл:

```
index.html
```

В приведенной команде ключ *--latest* требует взять за основу выгрузки в рабочий каталог последнее состояние, имеющееся в репозитории, а ключ *--force* вынуждает программу игнорировать тот факт, что в рабочем каталоге есть изменения, которые не были загружены в репозиторий, а значит, будут утеряны в результате выполнения команды. Если его не указать, то команда выполнена не будет, а будет выведено сообщение:

```
there are unsaved changes in the current checkout
```

4.2.6. Восстановление состояния

Заметим, что команда fossil checkout позволяет вернуть содержимое рабочего каталога не только к последнему состоянию репозитория, но и к любой другой точке сохранения, которая была создана в репозитории любым участником проекта при загрузке моментального снимка своего рабочего каталога. Узнать идентификатор необходимой точки сохранения можно не только через веб-интерфейс на странице шкалы времени, но и с помощью команды:

```
fossil timeline
```

Для рассматриваемого примера эта команда выведет строки:

```
=== 2020-03-04 ===
07:05:06 [38b8a0865f] *CURRENT* добавлен стилевой блок (user: Imposer2
tags: trunk)
=== 2020-03-02 ===
19:01:19 [d2bf34850e] index.html приведён к стандарту HTML5 (user:
Imposer2 tags: trunk)
07:13:30 [a8ebd7176c] Добавлен HTML-шаблон index.html (user: Imposer1
tags: trunk)
=== 2020-02-29 ===
07:24:50 [24e23c5cfb] initial empty check-in (user: SetupUser tags:
trunk)
+++ no more data (4) +++
```

В квадратных скобках приведены идентификаторы точек сохранения, за ними следуют их описания, а в конце в круглых скобках указаны имена пользователей Fossil — инициаторов соответствующих загрузок. Отметка *CURRENT* находится перед описанием точки сохранения, от моментального снимка кото-

рой ведут отсчёт изменения в файлах рабочего каталога, из которого была выполнена команда.

Чтобы вернуть рабочий каталог проекта в одно из хранящихся в репозитории состояний, надо ввести команду fossil checkout с указанием идентификатора желаемого состояния (точки сохранения), например:

```
fossil checkout d2bf34850e
```

После этого отметка *CURRENT* переместится на запись, ставшую текущей:

```
=== 2020-03-04 ===
07:05:06 [38b8a0865f] добавлен стилевой блок (user: Imposer2 tags:
trunk)
=== 2020-03-02 ===
19:01:19 [d2bf34850e] *CURRENT* index.html приведён к стандарту
HTML5 (user: Imposer2 tags: trunk)
07:13:30 [a8ebd7176c] Добавлен HTML-шаблон index.html (user: Imposer1
tags: trunk)
=== 2020-02-29 ===
07:24:50 [24e23c5cfb] initial empty check-in (user: SetupUser
tags: trunk)
+++ no more data (4) +++
```

Подтверждение факта восстановления состояния рабочего каталога можно получить с помощью команды fossil status:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02 19:01:19 UTC
parent: a8ebd7176c098f320a19a1c63e7b85c8adeb80fe 2020-03-02 07:13:30 UTC
child: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
tags: trunk
comment: index.html приведён к стандарту HTML5 (user: Imposer2)
```

Сейчас должны стать понятными значения полей checkout, parent и child, которые выводятся в отдельных строках блока информации о состоянии рабочего каталога проекта: checkout — это идентификатор текущей точки сохранения, от которой ведут отсчёт изменения в рабочем каталоге, parent — это идентификатор родительской точки сохранения, которая непосредственно предшествует текущей, а child — это идентификатор дочерней точки сохранения, которая непосредственно следует после текущей.

Почему же фактические значения идентификаторов, отображаемых командой fossil status, содержат гораздо больше шестнадцатеричных цифр, нежели значения, отображаемые в квадратных скобках командой timeline? Дело в том, что значения идентификаторов точек сохранения определяются результатом вычисления хеш-функций на сохраняемых состояниях. Это многозначные числа, но их характерная особенность заключается в том, что они сильно меняются во всех своих разрядах даже при незначительных изменениях состояний.

В большинстве случаев, возникающих на практике, различия присутствуют уже в первых десяти шестнадцатеричных разрядах. Поэтому ими ограничен вывод идентификаторов состояний в команде fossil timeline и в веб-интерфейсе на шкале времени.

На практике, когда требуется указать идентификатор состояния, можно не вводить даже все десять цифр его сокращённого значения. Достаточно указать первые разряды, позволяющие однозначно определить состояние. Например, для перехода из текущего состояния «index.html приведён к стандарту HTML5» к предшествующему состоянию «Добавлен HTML-шаблон index.html» можно ввести команду:

```
fossil checkout a8eb
```

При необходимости указываемое значение идентификатора состояния можно расширять и сверх десяти цифр — вплоть до его полной длины.

4.2.7. Конфронтация

А теперь предположим, что верстальщикам Imposer1 и Imposer2 не удалось договориться — каждый из них настаивает на своём варианте. Более того, верстальщик Imposer1 не последовал рекомендации системы Fossil получить из репозитория вариант Imposer2 и попытаться совместить исходные тексты (т. е. выполнить команду fossil update). Вместо этого он принудительно загрузил свои изменения в репозиторий командой:

```
fossil commit --allow-fork -m "к index.html подключён файл с таблицей
стилей index.css"
```

Команда завершилась успешно с выводом сообщений:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 397 received: 560 ip: 192.168.56.101
New_Version:
8da7a1351560cbb6d2419b652b75b713710b810082dc3c1b27b726e56a4e5fb8
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 3 received: 0
Sync done, sent: 1119 received: 671 ip: 192.168.56.101
***** WARNING: a fork has occurred *****
use "fossil leaves -multiple" for more details.
**** warning: a fork has occurred *****
```

Почему одно только использование опции *--allow-fork* позволило выполнить команду, ранее вызвавшую такое затруднение у системы контроля версий и потребовавшую серьёзного ручного вмешательства специалистов? Почему нельзя было поступить так же без дополнительных опций? И что означает дважды приведенная в сообщении команды строка «***** WARNING: a fork has occured *****» (ВНИМАНИЕ: произошло ответвление)?

Сначала посмотрим, в каком состоянии оказался рабочий каталог проекта с помощью команды fossil status:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 8da7a1351560cbb6d2419b652b75b713710b8100 2020-03-05 14:44:47 UTC
parent: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02 19:01:19 UTC
tags: trunk
comment: к index.html подключён файл с таблицей стилей index.css
(user: Imposer1)
WARNING: multiple open leaf check-ins on trunk:
(1) 2020-03-05 14:44:47 [8da7a13515] (current)
(2) 2020-03-04 07:05:06 [38b8a0865f]
```

Судя по тексту комментария, содержащемуся в поле *comment*, загрузка моментального снимка рабочего каталога в репозиторий состоялась. Вызывают вопрос последние три строки информационного блока. Сначала идёт предупреждение: несколько открытых загрузок в trunk. За ним приведены отметки времени двух загрузок с их идентификаторами в квадратных скобках. Первая из них отмечена маркером current (текущая).

Пояснить, что именно произошло в результате выполнения последней команды и в каком состоянии оказался репозиторий лучше всего на схеме временной шкалы проекта, доступной по гиперссылке http://fossil.server:8081/timeline (рис. 4.11).

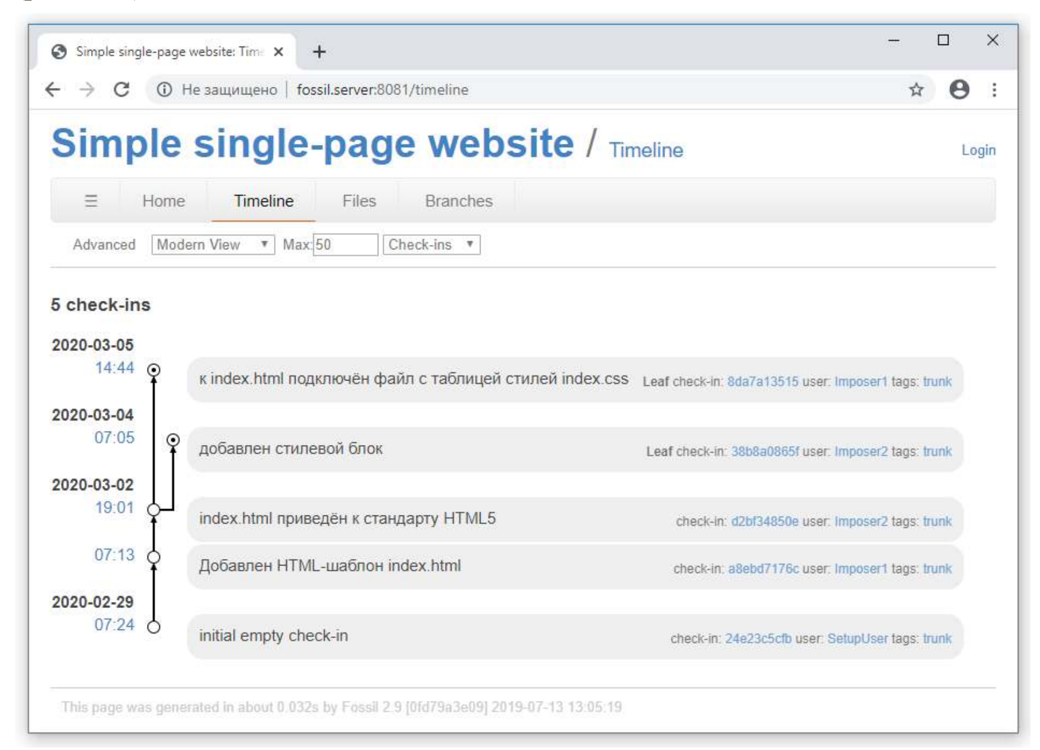


Рис. 4.11

Развилка в работе над проектом

На приведенном рисунке видно, что точка сохранения верстальщика Imposer2 с комментарием «добавлен стилевой блок» присутствует в репозитории, но она не получила дальнейшего развития. В то же время из точки сохранения, которая ей предшествует, выросла вторая ветвь, ведущая в только что созданную верстальщиком Imposer1 точку сохранения с комментарием «к index.html подключён файл с таблицей стилей index.css». То есть на данный момент в репозитории находятся два конкурирующих варианта развития проекта.

Так выглядит развилка в сетевом репозитории. С помощью команды fossil timeline можно взглянуть на неё из рабочего каталога:

```
=== 2020-03-05 ===
14:44:47 [8da7a13515] *CURRENT* к index.html подключён файл с таблицей
стилей index.css (user: Imposer1 tags: trunk)
=== 2020-03-04 ===
07:05:06 [38b8a0865f] добавлен стилевой блок (user: Imposer2 tags:
trunk)
=== 2020-03-02 ===
19:01:19 [d2bf34850e] *FORK* index.html приведён к стандарту HTML5
(user: Imposer2 tags: trunk)
07:13:30 [a8ebd7176c] Добавлен HTML-шаблон index.html (user: Imposer1
tags: trunk)
=== 2020-02-29 ===
07:24:50 [24e23c5cfb] initial empty check-in (user: SetupUser
tags: trunk)
+++ no more data (5) +++
```

Отметка *FORK* (РАЗВИЛКА) определяет точку сохранения, в которой произошло ответвление. Отметка *CURRENT* (ТЕКУЩАЯ) определяет текущую точку сохранения по отношению к рабочему каталогу.

А как идут дела у второго верстальщика? Он как раз синхронизировал репозитории командой fossil sync и, в свою очередь, тоже получил предупреждение о возникшей развилке:

```
Sync with http://Imposer2@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 3
Sync done, sent: 1486 received: 1674 ip: 192.168.56.101
***** WARNING: a fork has occurred *****
use "fossil leaves -multiple" for more details.
```

Поскольку он пока не в курсе последних событий, то решил воспользоваться рекомендацией системы Fossil и посмотреть подробности с помощью команды:

```
fossil leaves -multiple
```

Она вывела на экран строки с описаниями последних загрузок конкурирующих ветвей, возникших в результате развилки:

```
*** trunk ***
(1) 2020-03-05 14:44:47 [8da7a13515] к index.html подключён файл с
таблицей стилей index.css (user: Imposer1 tags: trunk)
(2) 2020-03-04 07:05:06 [38b8a0865f] добавлен стилевой блок (user:
Imposer2 tags: trunk)
```

Команда fossil status показала, что текущей точкой сохранения для рабочего каталога верстальщика Imposer2 является загрузка с комментарием «добавлен стилевой блок», о чём свидетельствует отметка current рядом с записью в перечне концевых узлов конкурирующих ветвей:

```
repository: …/Imposer2/sspwebsite/..\sspwebsite-local.fossil
local-root: …/Imposer2/sspwebsite/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04
07:05:06 UTC
parent: d2bf34850ef82c40430e31183664efd0501e348d 2020-03-02
19:01:19 UTC
tags: trunk
comment: добавлен стилевой блок (user: Imposer2)
WARNING: multiple open leaf check-ins on trunk:
(1) 2020-03-05 14:44:47 [8da7a13515]
(2) 2020-03-04 07:05:06 [38b8a0865f] (current)
```

Если сейчас не предпринять никаких мер, то проект будет развиваться в двух направлениях. Следующие загрузки верстальщиков будут записываться в параллельные ветки, всё дальше и дальше отдаляясь от точки, в которой произошла развилка, и тем самым уменьшая шансы достижения компромисса (рис. 4.12).

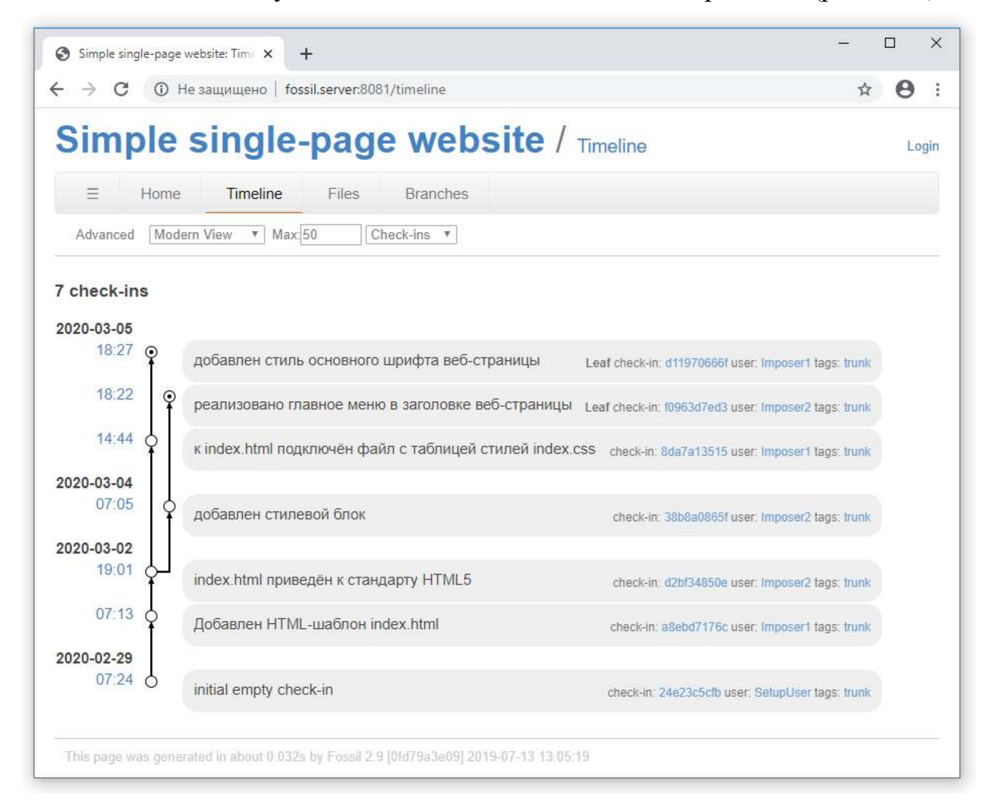


Рис. 4.12

Разработка пошла двумя параллельными путями

На рисунке видно, что загрузка верстальщика Imposer1 с описанием «добавлен стиль основного шрифта веб-страницы» попала в левую ветвь, а загрузка верстальщика Imposer2 с описанием «реализовано главное меню в заголовке веб-страницы» — в правую. Поскольку у нас не стоит задача получить два похожих продукта, то надо направить разработку в одно русло.

4.2.8. Разрешение конфликта

Когда разработчики не могут договориться между собой, в игру вступает начальство. Пришло время и менеджеру взглянуть на проект.

Первым делом Manager синхронизировал свой локальный репозиторий с сетевым с помощью команды fossil sync. После этого он посмотрел шкалу времени проекта с помощью команды fossil timeline. Вот что выдала эта команда:

```
=== 2020-03-05 ===
18:27:11 [d11970666f] добавлен стиль основного шрифта веб-страницы
(user: Imposer1 tags: trunk)
18:22:55 [f0963d7ed3] реализовано главное меню в заголовке веб-
страницы (user: Imposer2 tags: trunk)
14:44:47 [8da7a13515] к index.html подключён файл с таблицей сти-
лей index.css (user: Imposer1 tags: trunk)
=== 2020-03-04 ===
07:05:06 [38b8a0865f] добавлен стилевой блок (user: Imposer2 tags:
trunk)
=== 2020-03-02 ===
19:01:19 [d2bf34850e] *FORK* index.html приведён к стандарту HTML5
(user: Imposer2 tags: trunk)
07:13:30 [a8ebd7176c] *CURRENT* Добавлен HTML-шаблон index.html
(user: Imposer1 tags: trunk)
=== 2020-02-29 ===
07:24:50 [24e23c5cfb] initial empty check-in (user: SetupUser
tags: trunk)
+++ no more data (7) +++
```

Поскольку менеджер не выполнял никаких операций над своим рабочим каталогом, не обновлял его содержимое и не загружал в репозиторий изменённые файлы проекта, отметка *CURRENT* (ТЕКУЩИЙ) осталась на точке сохранения, моментальный снимок которой был использован при создании рабочего каталога.

Менеджер решил посмотреть, как выглядит проект в настоящий момент, для чего ввёл команду fossil checkout --latest. По этой команде система контроля версий переместила текущую отметку на последнюю по времени загрузку с идентификатором d11970666f (это результат работы верстальщика Imposer1) и выгрузила из репозитория в рабочий каталог файлы index.html и index.css, соответствующие ей.

Чтобы оценить работу верстальщика Imposer2, менеджер ввёл команду fossil checkout f0963d7ed3. По этой команде текущая отметка переместилась на загрузку с указанным идентификатором, и был выгружен соответствующий ей файл index.html в рабочий каталог. Поскольку второй верстальщик не использо-

вал внешний файл для хранения таблицы стилей, то файл index.css из рабочего каталога был удалён.

После просмотра обоих вариантов в веб-браузере менеджер пришёл к выводу, что проект ещё далёк от завершения, и он не может отдать предпочтение одному из них. Тогда он решил оценить объёмы проделанной работы каждым из верстальщиков, для чего запустил локальный веб-интерфейс командой fossil ui ..\sspwebsite-local.fossil и перешёл на шкалу времени по гиперссылке, представленной пунктом меню Timeline.

Менеджер переместил указатель мыши на кружок, обозначающий точку сохранения Imposer2 с комментарием «реализовано главное меню в заголовке веб-страницы», и щёлкнул левой кнопкой. Внутри кружка появилась красная метка. Затем он проделал то же самое с кружком, обозначающим точку сохранения Imposer1 с комментарием «добавлен стиль основного шрифта веб-страницы» (рис. 4.13).

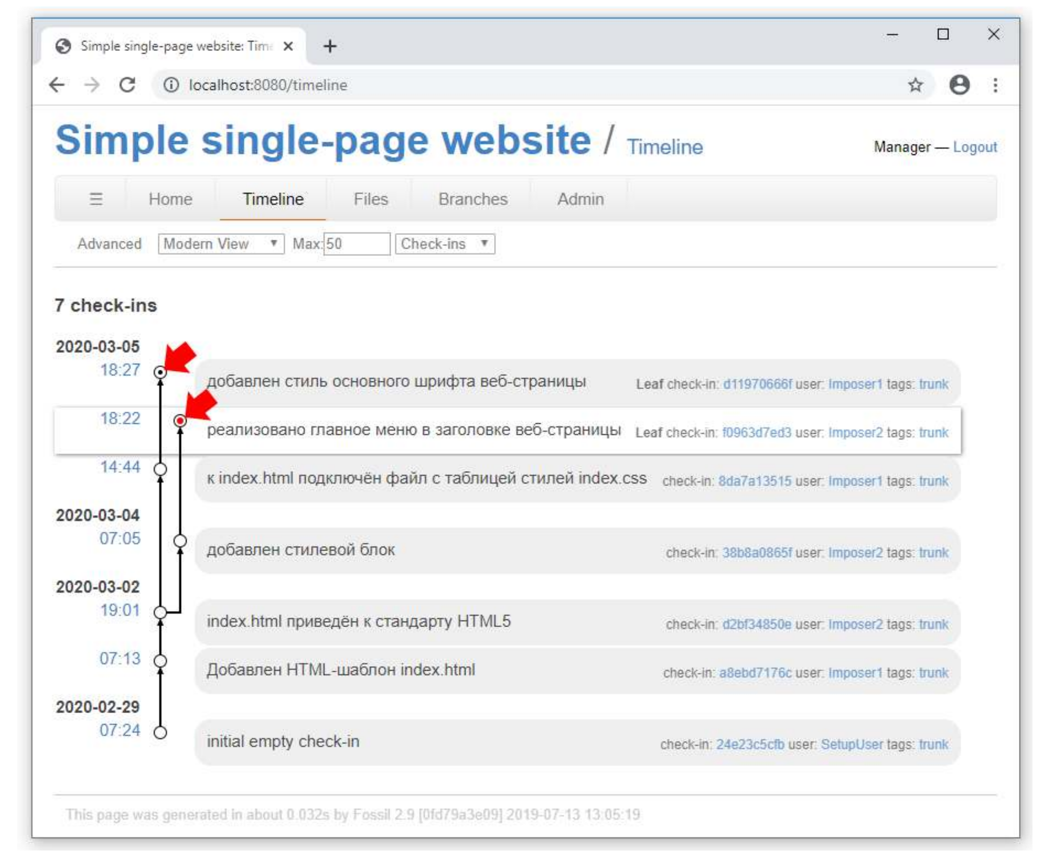


Рис. 4.13

Выполнение сравнения двух выгрузок

В веб-браузере отобразилась страница с наглядным представлением различий между выбранными точками сохранения (рис. 4.14). Менеджеру при-

шлось признать, что различия значительные и концептуальные, одной из ветвей разработки придётся пожертвовать.

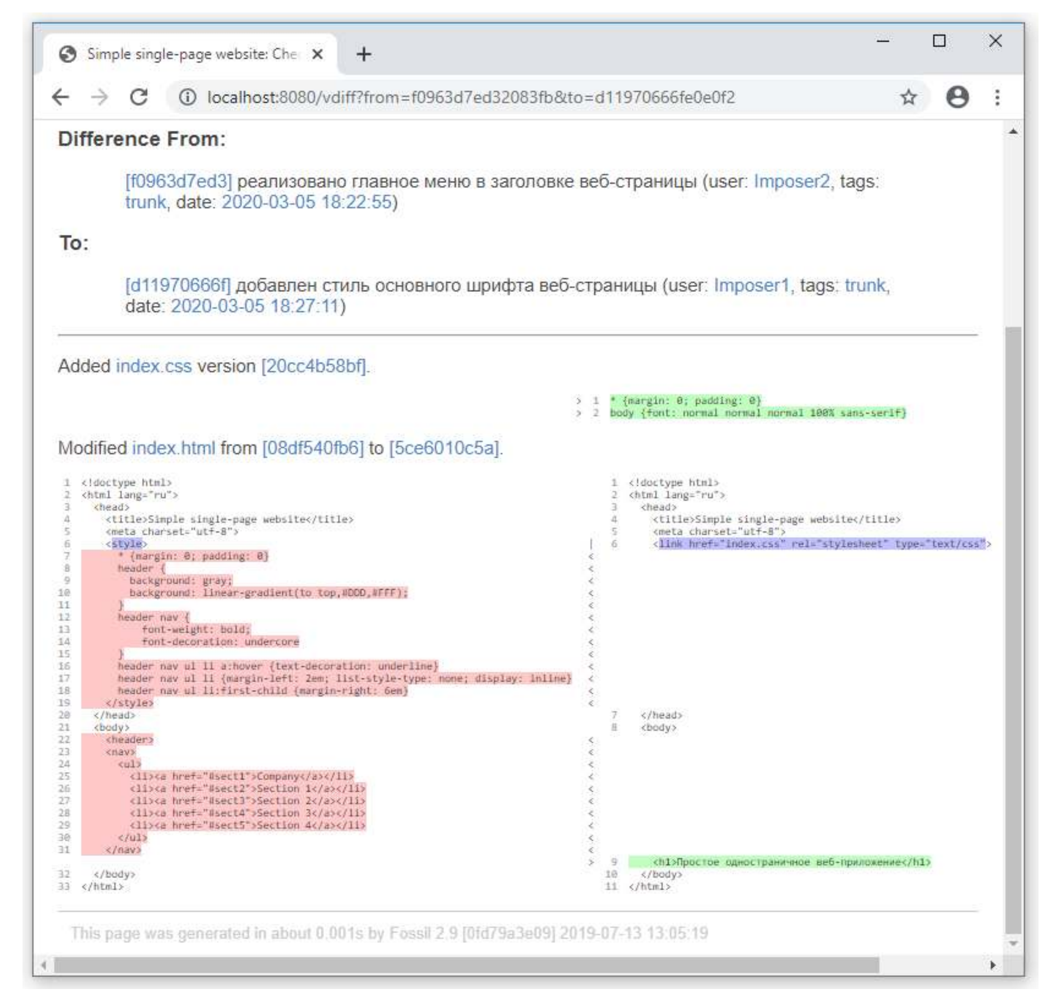


Рис. 4.14

Различия между конечными состояниями двух ветвей проекта

«Работа проделана большая, но так дело не пойдёт!» — подумал менеджер, ознакомившись с вариантом второго верстальщика. Идея размещения стилей в HTML-файле, пусть и в отдельном блоке, ему не понравилась. Совместить весь проект в одном файле всё равно не получится, потому что веб-сайт как минимум будет содержать графические элементы, а вынос параметров оформления в отдельный файл упростит как разработку, так и дальнейшее сопровождение.

В качестве основы для развития проекта менеджер выбрал вариант верстальщика Imposer1. Технически для этого ему понадобилось выполнить следующие действия.

Во-первых — сделать текущей ту точку сохранения, к которой будет приведён проект и от которой будет осуществляться его дальнейшее развитие. Это было выполнено командой:

```
fossil checkout d11970666f
```

где d11970666f — идентификатор точки сохранения.

Во-вторых — присоединить точку сохранения бесперспективной, по мнению менеджера, ветви к текущей точке сохранения:

```
fossil merge --integrate f0963d7ed3
```

В этой команде f0963d7ed3 — идентификатор точки сохранения присоединяемой ветви. Поскольку присоединяемую ветвь в дальнейшем развивать не планируется, то команда слияния дополнена опцией *--integrate* — она завершит присоединяемую ветвь, запретив в неё дальнейшие загрузки.

Для команды слияния fossil merge имеется ещё опция *--backout*, которая наряду с завершением присоединяемой ветви заставляет полностью игнорировать накопленные в ней с момента развилки изменения. Эту опцию можно было бы использовать, если бы решено было развивать вариант верстальщика Imposer2. Но в файле index.html, который предложил второй верстальщик, помимо неудачного решения с размещением оформления в стилевом блоке имеется интересная наработка относительно организации главного меню в заголовке веб-страницы, которую можно использовать в дальнейшем.

В результате выполнения операции слияния ветвей было выдано сообщение:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 361 received: 821 ip: 192.168.56.101
MERGE index.html
***** 1 merge conflicts in index.html
WARNING: 1 merge conflicts
"fossil undo" is available to undo changes to the working checkout.
```

Из сообщения понятно, что конфликт слияния ветвей произошёл в файле index.html. Открыв его в текстовом редакторе, менеджер увидел такую картину:

Файл index.html после попытки слияния

```
<!doctype html>
<html lang="ru">
  <head>
    <title>Simple single-page website</title>
    <meta charset="utf-8">
<<<<<<< BEGIN MERGE CONFLICT: local copy shown first <<<<<<<<<<<<<
    <link href="index.css" rel="stylesheet" type="text/css">
  </head>
======= COMMON ANCESTOR content follows ============================
  </head>
======= MERGED IN content follows ==================================
    <style>
```

```
      * {margin: 0; padding: 0}
      header {
        background: gray;
        background: linear-gradient(to top,#DDD,#FFF);
      }
      header nav {
          font-weight: bold;
          font-decoration: undercore
      }
      header nav ul li a:hover {text-decoration: underline}
      header nav ul li {margin-left: 2em; list-style-type: none; display: inline}
      header nav ul li:first-child {margin-right: 6em}
    </style>
  </head>
>>>>>>> END MERGE CONFLICT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  <body>
    <header>
      <nav>
        <ul>
          <li><a href="#sect1">Company</a></li>
          <li><a href="#sect2">Section 1</a></li>
          <li><a href="#sect3">Section 2</a></li>
          <li><a href="#sect4">Section 3</a></li>
          <li><a href="#sect5">Section 4</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

Строка BEGIN MERGE CONFLICT: local copy shown first отмечает начало текстового блока, в котором произошёл конфликт слияния и информирует о том, что непосредственно после неё идут строки из локальной копии файла — той, которая находилась в рабочем каталоге перед слиянием. И действительно, мы видим строку, ответственную за подключение файла index.css, содержащего стилевое оформление.

Строка COMMON ANCESTOR content follows отмечает начало текстового блока, содержащего строки общего предшественника для локального и вливаемого файлов.

Строка MERGED IN content follows содержит блок текста из вливаемого файла, который конфликтует (пересекается) с текстом, находящимся в локальной копии файла на момент слияния. Это как раз тот стилевой блок, который у второго верстальщика размещён на месте строки, подключающей недавно упомянутый файл index.css.

Строка END MERGE CONFLICT отмечает конец области конфликта. В оставшейся части текстового файла находятся строки, ради которых проводилось слияние — ответственные за формирование горизонтального меню в заголовке веб-страницы.

Ознакомившись со структурой файла index.html, сформированного системой Fossil при попытке совмещения двух вариантов, он сделал следующее:

1) перенёс описания стилей из стилевого блока в файл index.css;

2) удалил из index.html строки, содержащиеся между отметками COMMON ANCESTOR content follows и END MERGE CONFLICT вместе с самими отметками;

3) удалил строку с отметкой BEGIN MERGE CONFLICT: local copy shown first;

4) удалил альтернативные варианты файлов index.html, созданные в рабочем каталоге системой Fossil, с помощью команды del *-* (на системах UNIX надо выполнить команду rm *-*).

В результате файл index.html приобрёл вид:

Файл index.html после ручной доработки

```
<!doctype html>
<html lang="ru">
  <head>
    <title>Simple single-page website</title>
    <meta charset="utf-8">
    <link href="index.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <header>
      <nav>
        <ul>
          <li><a href="#sect1">Company</a></li>
          <li><a href="#sect2">Section 1</a></li>
          <li><a href="#sect3">Section 2</a></li>
          <li><a href="#sect4">Section 3</a></li>
          <li><a href="#sect5">Section 4</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

А файл index.css стал выглядеть так:

Файл index.css после ручной доработки

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header {background: gray; background: linear-gradient(to top,#DDD,#FFF)}
header nav {font-weight: bold; font-decoration: undercore}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; display: inline}
header nav ul li:first-child {margin-right: 6em}
```

После выполненных изменений команда fossil status показывает такой результат:

```
repository: …/Manager/website/..\sspwebsite-local.fossil
local-root: …/Manager/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: d11970666fe0e0f2863903df419f8a2b0c5283b1 2020-03-05 18:27:11 UTC
parent: 8da7a1351560cbb6d2419b652b75b713710b8100 2020-03-05 14:44:47 UTC
```

```
tags: trunk
comment: добавлен стиль основного шрифта веб-страницы (user: Imposer1)
EDITED index.css
EDITED index.html
INTEGRATE
f0963d7ed32083fbca7903f8b6aad49a18e1584f2c66ebee7f6dfb35ca575111
WARNING: multiple open leaf check-ins on trunk:
(1) 2020-03-05 18:27:11 [d11970666f] (current)
(2) 2020-03-05 18:22:55 [f0963d7ed3]
```

Третье, и последнее, что осталось сделать менеджеру для решения конфликта, — загрузить результат проделанных операций по слиянию ветвей проекта в репозиторий:

```
fossil commit -m "выбрано направление дальнейшего развития проекта"
```

После вмешательства менеджера шкала времени сетевого репозитория приобрела вид, показанный на рисунке 4.15.

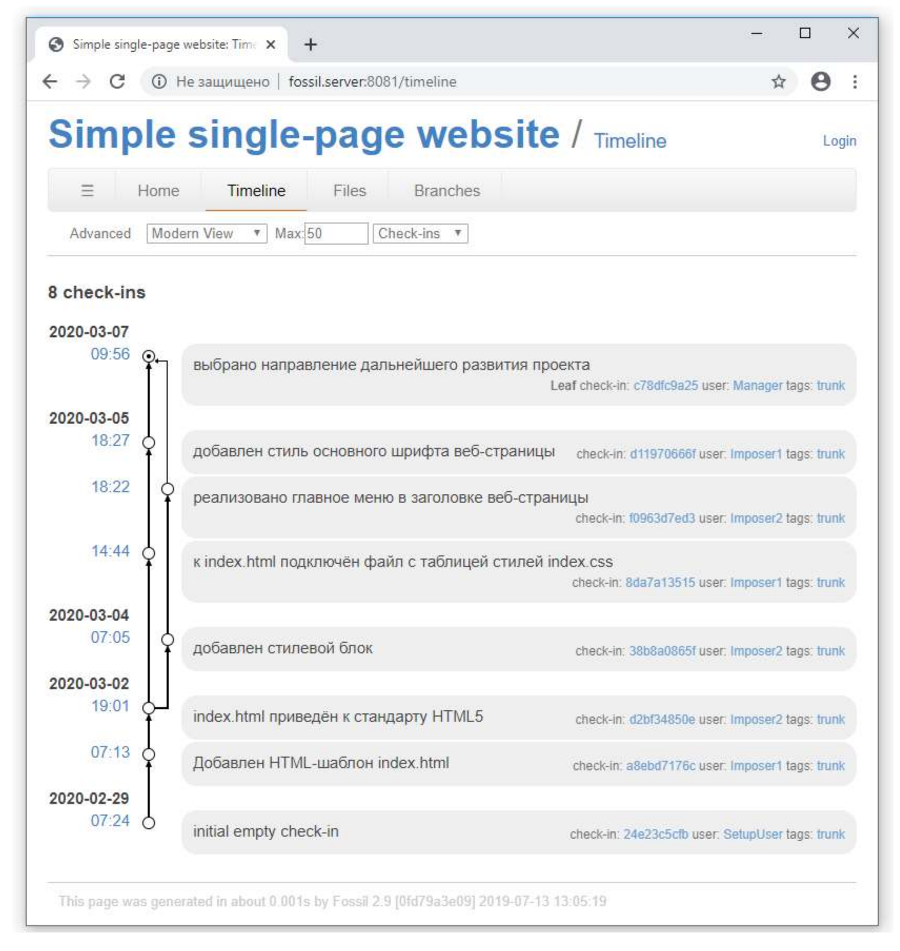


Рис. 4.15

Шкала времени после слияния ветвей проекта

Эту же шкалу времени можно увидеть в окне командной строки, если воспользоваться командой fossil timeline:

```
=== 2020-03-07 ===
09:56:15 [c78dfc9a25] *MERGE* *CURRENT* выбрано направление дальнейшего развития проекта (user: Manager tags: trunk)
=== 2020-03-05 ===
18:27:11 [d11970666f] добавлен стиль основного шрифта веб-страницы (user: Imposer1 tags: trunk)
18:22:55 [f0963d7ed3] реализовано главное меню в заголовке веб-страницы (user: Imposer2 tags: trunk)
14:44:47 [8da7a13515] к index.html подключён файл с таблицей стилей index.css (user: Imposer1 tags: trunk)
=== 2020-03-04 ===
07:05:06 [38b8a0865f] добавлен стилевой блок (user: Imposer2 tags: trunk)
=== 2020-03-02 ===
19:01:19 [d2bf34850e] *FORK* index.html приведён к стандарту HTML5 (user: Imposer2 tags: trunk)
07:13:30 [a8ebd7176c] Добавлен HTML-шаблон index.html (user: Imposer1 tags: trunk)
=== 2020-02-29 ===
07:24:50 [24e23c5cfb] initial empty check-in (user: SetupUser tags: trunk)
+++ no more data (8) +++
```

Отметка *FORK* помогает понять, где произошло разветвление проекта, а отметка *MERGE* — где произошло слияние.

4.2.9. Альтернативное разрешение

К этому моменту читатель, вероятно, уже недоумевает по поводу того, какой значительный объём технической работы проделал менеджер для решения конфликта, возникшего между верстальщиками. На самом деле у менеджера хватает дел по управлению проектом и не всегда есть достаточная квалификация, чтобы компетентно править исходный код.

Но в рамках этой книги предположим, что менеджер — в недавнем прошлом лидер команды разработчиков, решивший попробовать себя в новом статусе. Тем более что проект небольшой, и в возникшем конфликте он нашёл для себя возможность тряхнуть стариной. Поэтому отложим организационные вопросы, а сосредоточимся на технической стороне дела.

Более того, предположим, что в параллельной вселенной менеджер после анализа ситуации вокруг конфликта стал на сторону верстальщика Imposer2 и решил пожертвовать работой первого верстальщика, потому что тот набрал всего несколько строк. Вот что предпринял менеджер в этом случае.

Во-первых, он сделал текущей последнюю точку сохранения верстальщика Imposer1 с помощью команды:

```
fossil checkout f0963d7ed3
```

где f0963d7ed3 — идентификатор точки сохранения.

Во-вторых, он присоединил последнюю точку сохранения ветви верстальщика Imposer2 к текущей точке сохранения с указанием отбросить все изменения, выполненные в этой ветви с момента развилки. Для этого менеджер дополнил команду слияния опцией *--backout*:

```
fossil merge --backout d11970666f
```

где d11970666f — идентификатор точки сохранения прерываемой ветви.

В-третьих, он загрузил результат проделанных операций по слиянию ветвей проекта в репозиторий. Поскольку в файлы рабочего каталога никаких изменений не вносилось, команду выгрузки следует дополнить опцией *--allow-empty*:

```
fossil commit --allow-empty -m "выбрано направление дальнейшего
развития проекта"
```

После вмешательства менеджера шкала времени сетевого репозитория приобрела вид, показанный на рисунке 4.16. Пунктирной линией обозначена фиктивная загрузка, не содержащая изменённых файлов, но отражающая слияние ветвей проекта.

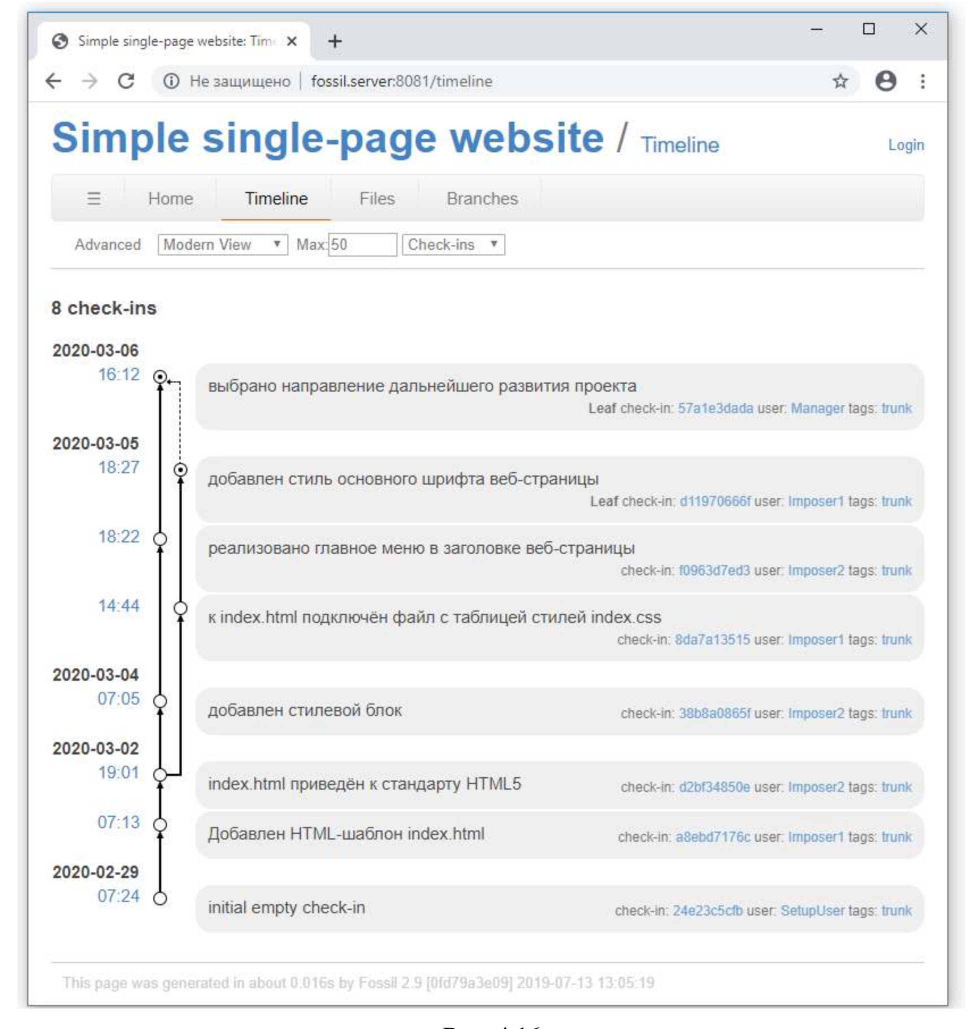


Рис. 4.16

Шкала времени после слияния конкурирующих ветвей

После слияния двух параллельных ветвей проекта его участникам следует выполнить обновление своих локальных репозиториев (с помощью команды fossil sync или fossil pull) и рабочих каталогов (с помощью команды fossil update). При коллективной разработке эти действия лучше производить более-менее регулярно, чтобы иметь актуальное состояние проекта на своём рабочем месте. Это уменьшит вероятность наложений несовместимых изменений, хотя, конечно, не сможет полностью их исключить. Как решать возникающие при этом конфликты — в общих чертах было рассмотрено выше.

На этом рассмотрение примера возникновения и решения конфликта завершено. Пришло время подвести итоги и систематизировать полученные сведения.

4.2.10. Уроки конфликта

На первый взгляд может показаться, что решение конфликтов, возникших в ходе разработки, через систему контроля версий требует слишком больших усилий. Небольшие команды, которые до сих пор не пользуются специализированными системами, а просто держат исходные тексты в общедоступном файловом хранилище, могут заявить, что подобные конфликты у них исключены договорённостями и соглашениями, действующими среди участников проекта. Работа же через систему контроля версий несёт с собой большие накладные расходы в виде ритуалов синхронизации репозиториев и постоянного контроля над текущим состоянием проекта.

Такие заявления кажутся справедливыми лишь на первый взгляд. Даже если работа над проектом распределена между его участниками так, чтобы исключить изменение одних и тех же файлов разными разработчиками, система контроля версий станет жёстким каркасом, направляющим разработку. Кроме того, возникновение отдельных инцидентов нельзя исключать, и если без системы контроля версий один разработчик может просто переписать файл с результатами работы своего коллеги, то в системе контроля версий этот инцидент будет незамедлительно обнаружен, а усилия по управляемому решению возникшего конфликта, скорее всего, предотвратят крупные проблемы, которые могли бы возникнуть в дальнейшем.

Кроме гарантирования целостности проекта в ходе разработки система контроля версий обеспечивает хранение всей истории его изменений. С помощью команды fossil checkout можно получить все файлы проекта и в том виде, в каком они существовали на момент создания точки сохранения. Никто не запрещает создать несколько рабочих каталогов, открыть из них один и тот же файл локального репозитория с помощью команды fossil open и выгрузить в эти рабочие каталоги из репозитория файлы проекта, соответствующие разным точкам сохранения. Такая модель использования позволяет получить полное представление о состоянии проекта на разных этапах его исторического развития или одновременно вести разработку нескольких параллельных ветвей (например, для проверки нескольких возможных вариантов реализации нетривиальной функциональности).

Обнаружение при выгрузке актуального состояния репозитория в свой рабочий каталог командой fossil update перекрывающихся изменений факта, что файл, над которым велась работа, одновременно был модифицирован другим участником проекта, — момент неприятный, но не трагичный. Система Fossil делает всё возможное, чтобы упростить консолидацию конфликтующих версий.

Во-первых, она создаёт в рабочем каталоге три варианта файла, вызвавшего проблему:

- с суффиксом -original — локальный файл, который присутствовал в рабочем каталоге на момент получения обновления;
- с суффиксом -merge — файл из репозитория, который изменён другим участником проекта;
- с суффиксом -baseline — файл, содержащий лишь общие для предыдущих двух вариантов части.

Во-вторых, она аккуратно вставляет конфликтующие блоки в нужные места локального файла, сохраняя их последовательность и снабжая комментариями. Как правило, для решения конфликта разработчику остаётся просмотреть эти участки, продумать план консолидации и исправить файл, учитывая интересы участников конфликта.

В большинстве случаев система Fossil позволяет откатить изменения в рабочем каталоге, вызванные выполнением последней команды. Для этого надо ввести команду:

```
fossil undo
```

Например, верстальщик Imposer2 находится ещё в неведении относительно решения, принятого по поводу его варианта работы, о чём говорит команда fossil status:

```
repository: …/Imposer2/sspwebsite/..\sspwebsite-local.fossil
local-root: …/Imposer2/sspwebsite/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: f0963d7ed32083fbca7903f8b6aad49a18e1584f 2020-03-05 18:22:55 UTC
parent: 38b8a0865f8e4642cec82705f51b4d564dac0ca2 2020-03-04 07:05:06 UTC
merged-into: c78dfc9a256c1bc2e02692a0fd6b2014071bae53 2020-03-07 09:56:15 UTC
tags: trunk
comment: реализовано главное меню в заголовке веб-страницы (user: Imposer2)
```

Но при очередном получении обновлений проекта из репозитория в рабочий каталог с помощью команды fossil update он увидел такие сообщения:

```
Autosync: http://Imposer2@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 396 received: 932 ip: 192.168.56.101
ADD index.css ДОБАВЛЕН index.css
UPDATE index.html ОБНОВЛЁН index.html
-------------------------------------------------------------
updated-to: c78dfc9a256c1bc2e02692a0fd6b2014071bae53 2020-03-07 09:56:15 UTC
tags: trunk
```

```
comment: выбрано направление дальнейшего развития проекта (user: Manager)
changes: 2 files modified. 2 файла изменены.
"fossil undo" is available to undo changes to the working checkout.
```

И действительно, в рабочем каталоге появится файл index.css, а файл index.html будет представлен в том варианте, каким его загрузил в репозиторий менеджер при решении конфликта. Если по какой-то причине результат загрузки обновлений верстальщика не устраивает, он может выполнить команду:

```
fossil undo
```

На экран будут выведены сообщения:

```
DELETE index.css УДАЛЁН index.css
UNDO index.html ВОССТАНОВЛЕН index.html
--------------------
updated-to: f0963d7ed32083fbca7903f8b6aad49a18e1584f 2020-03-05 18:22:55 UTC
tags: trunk
comment: реализовано главное меню в заголовке веб-страницы (user: Imposer2)
```

После отката рабочий каталог будет возвращён в состояние, которое предшествовало выполнению команды fossil update.

Если при выполнении команды fossil update система обнаружит, что репозиторий не содержит ничего, что отличалось бы от содержимого рабочего каталога, то в строке changes (изменения) будет выведено сообщение: None. Already up-to-date (Ничего. Уже актуально).

Уже отмечалось ранее, но будет не лишним напомнить, что работа с сетевым репозиторием осуществляется опосредовано, через локальный репозиторий. Команда fossil sync выполняет двухстороннюю синхронизацию локального и сетевого репозиториев:

- получает изменения из сетевого репозитория в локальный (fossil pull);
- отправляет изменения из локального репозитория в сетевой (fossil push).

Выполнение синхронизации никак не влияет на состояние рабочего каталога. Информационный обмен между рабочим каталогом и репозиторием инициируется другими командами:

- fossil commit — загрузка мгновенного снимка рабочего каталога в репозиторий с созданием в последнем точки сохранения;
- fossil update — выгрузка актуального состояния из репозитория в рабочий каталог с наложением на него выполненных локально, но ещё не загруженных в репозиторий изменений;
- fossil checkout — выгрузка сохранённого состояния из репозитория в рабочий каталог с утратой выполненных локально, но ещё не загруженных в репозиторий изменений.

Установленный по умолчанию режим автоматической синхронизации приводит к тому, что выполнение команд информационного обмена между рабочим каталогом и репозиторием запускает процедуру синхронизации локального репозитория с сетевым.

4.3. Параллельная разработка

4.3.1. Разветвления и ответвления

Из рассмотренного примера может сложиться впечатление, что разветвление линии развития проекта — это что-то нежелательное, вызванное какими-то несовместимыми изменениями и требующее арбитража. Так и есть, появление ветви из-за невозможности бесконфликтного слияния изменений, внесённых в файлы проекта его участниками, — это форс-мажор, который надо как можно быстрее устранить. Для такой ситуации в системе контроля версий используется термин fork.

Но в реальной практике нередки случаи, когда участник проекта может по собственному желанию создать ответвление от главной линии развития. Причин тому может быть несколько:

- одновременная разработка нескольких функций;
- отсутствие уверенности в правильности изменений, которые предстоит внести;
- испытание нескольких вариантов реализации одной и той же функции;
- тестирование программного продукта перед выпуском;
- сопровождение выпущенной в промышленную эксплуатацию версии программного продукта.

Коллективная работа над проектом подразумевает разделение труда. Лучше всего, если есть возможность выдать участникам проекта независимые задания, над которыми они могли бы работать одновременно. Прекрасно, если каждый разработчик при этом будет вносить изменения только в свои наборы файлов. Отразить ведение параллельной разработки в системе контроля версий можно созданием для каждого задания отдельной ветви. По мере выполнения заданий наработки из ветвей вливаются в основную ветвь проекта, постепенно доводя его до готовности.

Если разработчик заранее не уверен в правильности выбранного направления развития проекта, то лучше сразу зафиксировать это сомнение путём вывода дальнейшей разработки в отдельную ветку. Такой подход упростит возврат к точке сомнения в случае, если подтвердятся негативные ожидания. А если сомнения окажутся напрасными, то не составит никакого труда внедрить результаты работы в рамках созданной ветки в основную ветвь проекта.

Одним из вариантов отсутствия уверенности может быть испытание в работе над реальным проектом практиканта или новичка команды. Лучший вариант введения в курс дела нового человека — предложить ему выполнить конкретное задание. А чтобы работа над ним не помешала остальным членам команды, для реализации этого задания можно создать отдельную ветвь в репозитории. В случае успеха реализованная функция будет принята в основную ветвь, а в случае неудачи можно попытаться разобраться с её причинами, проанализировав точки сохранения выделенной ветви.

Некоторые функции могут быть реализованы несколькими способами. Простейший пример — сортировка массива данных. Разные методы сортировки

могут показать различную производительность в зависимости от вида исходных данных. Если это является принципиальным, то можно реализовать несколько методов, а потом протестировать работу системы с каждым из них и выбрать наиболее эффективный. Работу над каждым альтернативным методом удобнее всего вести в отдельной ветви проекта.

В какой-то момент разработки принимается решение о том, что продукт созрел для выпуска в эксплуатацию. На этом этапе фиксируется набор функций, которые делает программа, и выполняется только шлифовка с устранением ошибок. В то же время выпускаемая версия может быть не окончательной, если имеются планы относительно дальнейшего развития продукта. В этом случае уместно выделить работу над ней в отдельную ветку, а разработку перспективных функций для следующих выпусков продолжать в основной ветви.

В выпущенных программах почти всегда обнаруживаются недостатки. Если не было создано ответвления перед выпуском продукта, разумно его создать в случае появления необходимости его доработки после выпуска. В любом случае лучше отделить работу над программой с зафиксированной функциональностью от работы над продолжением развития программного продукта.

Все рассмотренные примеры объединяет одно: решение о создании ветки в разработке проекта принимается осознанно и не для устранения последствий аварии, а превентивно, для улучшения управляемости процесса разработки. Ветви разработки, которые возникают как в результате таких ответвлений, так и в результате разветвлений, произошедших из-за форс-мажорных обстоятельств, в системе контроля версий обозначают термином branch. Несмотря на то, что причины их происхождения различны, дальнейшая работа с ветвями производится одинаково.

4.3.2. Создание ответвлений

После решения конфликта, который, как мы помним, произошёл из-за различных взглядов верстальщиков на способ подключения стилевого оформления к веб-сайту, менеджер проекта решил принять организационные меры для уменьшения вероятности таких накладок в дальнейшем. Он предложил верстальщику Imposer2, который начал разработку главного меню в заголовке веб-сайта, продолжить работу и заняться заголовком страницы в целом. А верстальщику Imposer1 предложил заняться основным содержимым веб-сайта.

Узнав об отсутствии у верстальщиков возражений относительно распределения задач, менеджер решил отразить принятое решение в репозитории проекта путём создания двух ветвей — по одной для каждой поставленной задачи. Прежде всего ему понадобилось узнать идентификатор точки сохранения, в которой предстоит сделать запланированные ответвления. Он воспользовался командой:

```
fossil timeline -n 1
```

С помощью параметра *-n* менеджер ограничил выдачу одной последней записью и получил интересующие его сведения:

```
=== 2020-03-07 ===
```

```
09:56:15 [c78dfc9a25] *MERGE* *CURRENT* выбрано направление даль-
нейшего развития проекта (user: Manager tags: trunk)
--- entry limit (1) reached --- достигнут предел записей (1)
```

Теперь в точке сохранения с идентификатором c78dfc9a25 можно создать ответвление для работы над заголовком веб-страницы:

```
fossil branch new заголовок c78dfc9a25
```

В выполненной команде заголовок — это имя для создаваемой ветви. При разветвлениях, которые происходят в результате конфликтов, возникающие ветви обычно безымянны. Если же ответвление производится преднамеренно, то уместно назвать создаваемую ветвь так, чтобы было понятно её назначение. В результате выполнения команды на экран была выведена следующая информация:

```
New branch: 93b91744b73976e1c25811deeeb1ac5982ab66b46a9afb4fd11524423a4d8d7d
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 1 received: 0
Push done, sent: 1290 received: 281 ip: 192.168.56.101
```

В первой строке указан идентификатор созданной ветви, он же — идентификатор её первой точки сохранения. Остальные строки содержат служебную информацию о произошедшей при выполнении команды синхронизации репозиториев. Аналогичным образом создаётся вторая ветвь — для работы над содержимым веб-страницы:

```
fossil branch new содержимое c78dfc9a25
```

Полученный ответ такой же, как в результате выполнения первой команды, за исключением фактических значений:

```
New branch: ddc308408924cddce8a39c2f79f51270bad443942753e6541bc59f0e6f6f33a7
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 1 received: 0
Push done, sent: 1332 received: 281 ip: 192.168.56.101
```

После создания ответвлений на шкале времени проекта в дополнение к основной ветви, которая называется trunk, появились ветви «заголовок» и «содержимое» (рис. 4.17).

При преднамеренном ответвлении с помощью команды fossil branch new происходит копирование точки сохранения, из которой вырастает ветвь, в точку сохранения, которой новая ветвь начинается. В приведенном примере точкам сохранения с идентификаторами c78dfc9a25, 93b91744b7 и ddc3084089 соответствуют одинаковые моментальные снимки рабочего каталога.

Несмотря на это, участники проекта должны переключиться на ту ветвь, которую они будут развивать. Верстальщик Imposer1 выполнил команду fossil update, чтобы актуализировать состояние своего рабочего каталога, и посмотрел последние три события в истории развития проекта с помощью команды:

```
fossil timeline -n 3
```

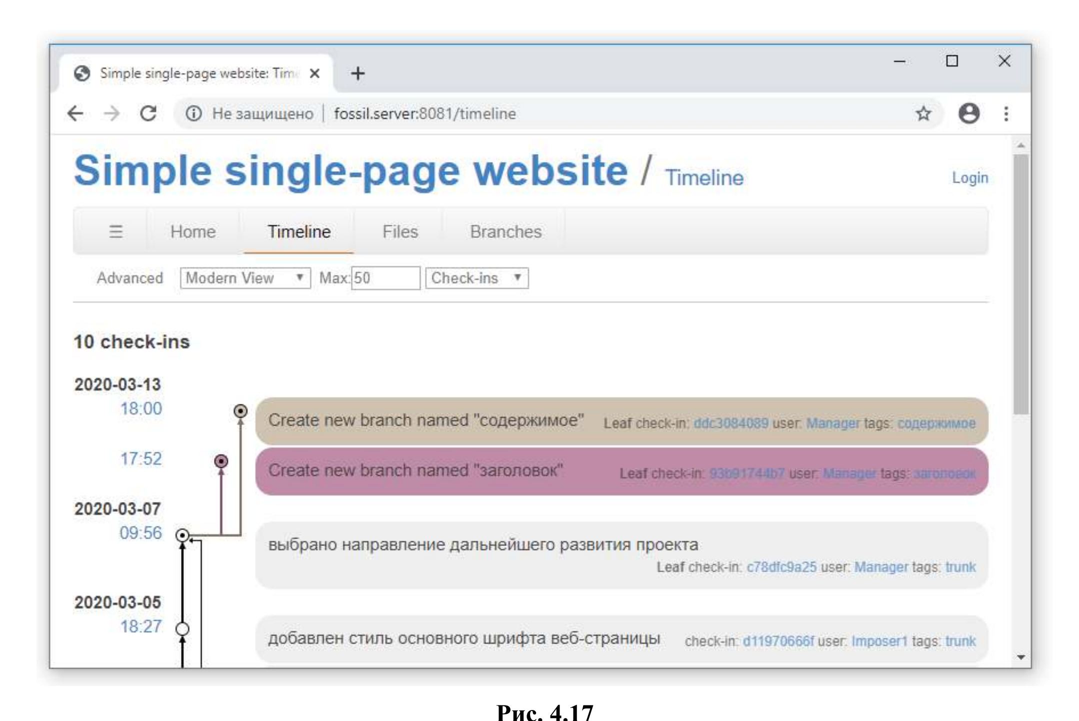


Рис. 4.17

Созданные ветви «заголовок» и «содержимое» на шкале времени

Это позволило ему узнать идентификатор ветви, предназначенной для работы над содержимым веб-страницы:

```
=== 2020-03-13 ===
18:00:58 [ddc3084089] Create new branch named "содержимое" (user:
Manager tags: содержимое)
17:52:44 [93b91744b7] Create new branch named "заголовок" (user:
Manager tags: заголовок)
=== 2020-03-07 ===
09:56:15 [c78dfc9a25] *MERGE* *BRANCH* *CURRENT* выбрано направле-
ние дальнейшего развития проекта (user: Manager tags:
trunk)
--- entry limit (3) reached --- достигнут предел записей (3)
```

Чтобы переключиться на ветвь «содержимое», первый верстальщик выполнил команду выгрузки из репозитория в свой рабочий каталог моментального снимка, соответствующего первой точке сохранения интересующей его ветви:

```
fossil checkout ddc3084089
```

В ходе выполнения команда сообщила о выгрузке из репозитория двух файлов:

```
index.css
index.html
```

Чтобы убедиться, что переключение на нужную ему ветвь произошло, верстальщик ввел команду:

```
fossil branch
```

На экране отобразился список ветвей проекта, в котором активная ветвь отмечена символом *:

```
trunk
заголовок
* содержимое
```

Таким образом, верстальщик Imposer1 подготовился к работе над содержимым веб-страницы.

В это же время второй верстальщик посмотрел на шкалу времени проекта через веб-интерфейс и узнал, что идентификатор ветви «заголовок», которую ему предстоит развивать, равен 93b91744b7. Чтобы переключиться на эту ветвь, он ввёл команду:

```
fossil checkout 93b91744b7
```

Но вместо переключения команда вывела сообщение об ошибке:

```
not found: 93b91744b7 не найдено: 93b91744b7
```

Так произошло потому, что в локальном репозитории пользователя Imposer2 пока ещё отсутствуют сведения о том, что в проекте были созданы ответвления. Убедиться в этом можно с помощью команды fossil timeline -n 1:

```
=== 2020-03-07 ===
09:56:15 [c78dfc9a25] *MERGE* *CURRENT* выбрано направление даль-
нейшего развития проекта (user: Manager tags: trunk)
--- entry limit (1) reached ---
```

Поскольку второй разработчик не изменял файлы в рабочем каталоге с момента последней синхронизации (в этом он убедился, введя команду fossil changes и получив пустой ответ), он решил не выполнять полную синхронизацию, а только получить последние обновления из сетевого репозитория в локальный, для чего выполнил команду:

```
fossil pull
```

Команда сообщила об успешном получении сведений:

```
Pull from http://Imposer2@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 2
Pull done, sent: 891 received: 2337 ip: 192.168.56.101
```

После того, как в локальный репозиторий попали имеющиеся в сетевом репозитории сведения о созданных менеджером ветвях, команда переключения на ветвь «заголовок» отработала в штатном режиме, сообщив о выгрузке из репозитория в рабочий каталог двух файлов: index.css и index.html.

Чтобы убедиться в успешном переключении, верстальщик Imposer2 ввёл команду fossil branch и получил подтверждение:

```
trunk
* заголовок
содержимое
```

Можно отметить, что создание новых ветвей и переключение на них участников проекта происходили не совсем легко из-за того, что требовалось узнавать идентификаторы точек сохранения для использования в командах. Но процесс значительно упрощается, если знать, что название ветви может использоваться в качестве синонима идентификатора последней точки сохранения, находящейся на ней. С учётом этого команды создания новых ветвей, выполняемые менеджером, могут быть записаны в виде:

```
fossil branch new заголовок trunk
fossil branch new содержимое trunk
```

Поскольку trunk — это название основной ветви разработки, то за основу создания новых ветвей будет использована последняя точка сохранения на ней, а это приведёт к тому же результату, что и при выполнении команд, указанных выше. Естественно, такой способ не сработает, если за основу новой ветви надо взять не последнюю, а более раннюю точку сохранения.

Помимо этого, для переключения на новые ветви разработчики могут использовать такие команды:

- fossil checkout содержимое — выполняется первым верстальщиком для переключения на ветвь «содержимое»;
- fossil checkout заголовок — выполняется вторым верстальщиком для переключения на ветвь «заголовок».

О том, почему возможна замена идентификаторов точек сохранения названиями ветвей, будет подробно рассказано в разделе, посвящённом ярлыкам и свойствам.

А дальше работа над проектом пошла своим чередом. Руководство приняло решение использовать разрабатываемый в рамках проекта одностраничный веб-сайт в качестве портфолио и выложить его под свободной лицензией MIT. Менеджер создал в своём рабочем каталоге файл LICENSE, заполнил его по образцу, размещённому на сайте https://mit-license.org/, и загрузил в репозиторий, выполнив последовательность команд:

```
fossil add LICENSE
fossil commit -m "добавлен файл лицензии MIT"
```

Первый верстальщик в секции body файла index.html создал секцию main, которой обозначил свой фронт работ, и начал разработку структуры блока section. Когда промежуточный результат его удовлетворил, он загрузил его в репозиторий с помощью команды:

```
fossil commit -m "прототип блока содержимого - main, section"
```

Второй верстальщик в уже существующей секции header файла index.html создал блок div для названия организации и её девиза, после чего загрузил свою работу в репозиторий, выполнив команду:

```
fossil commit -m "блок названия организации и её девиза"
```

После этого шкала времени проекта стала выглядеть, как показано на рисунке 4.18.

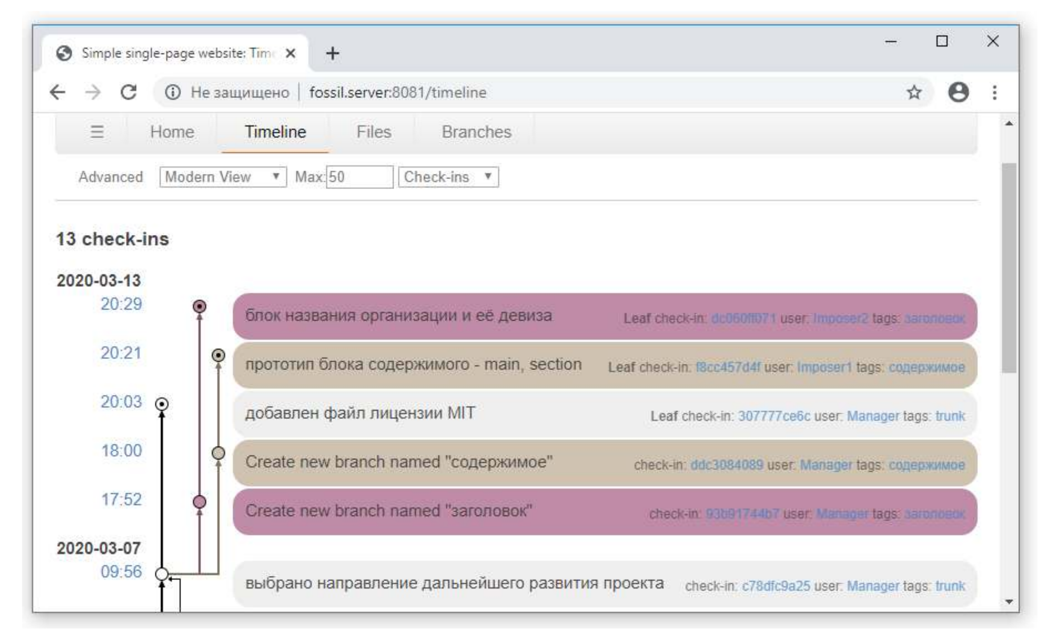


Рис. 4.18

Параллельное развитие трёх ветвей проекта

4.3.3. Перемещение файлов в рабочем каталоге

В это же время дизайнер закончил работу над графическими элементами веб-сайта и готов предоставить файлы изображений верстальщикам. Он — новичок в использовании репозитория и вынужден пройти те шаги, которые его коллеги уже давно выполнили (рис. 4.19). Посмотрим, что он делает, а заодно вспомним, как начинается работа с репозиторием.

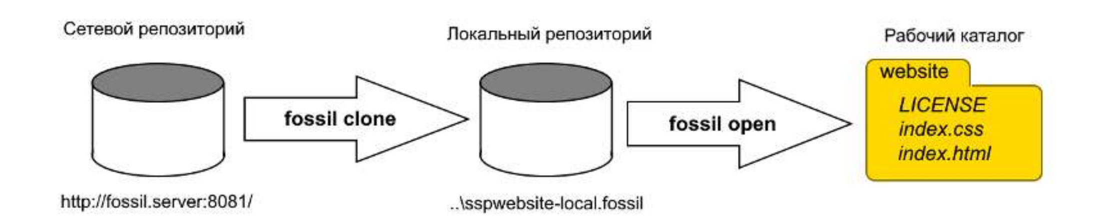


Рис. 4.19

Схема подготовки к работе с репозиторием проекта

Сначала дизайнер пытается создать локальную копию сетевого репозитория на своём компьютере:

```
fossil clone http://designer@fossil.server/ sspwebsite-local.fossil
```

Он вводит пароль, подтверждает его сохранение, но команда завершается с ошибкой:

```
password for designer: ********
remember password (Y/n)?
Round-trips: 2 Artifacts sent: 0 received: 26
Error: login failed Ошибка: авторизация не выполнена
Round-trips: 2 Artifacts sent: 0 received: 26
Clone done, sent: 567 received: 9750 ip: 192.168.56.101
server returned an error - clone aborted сервер вернул ошибку - клонирование прервано
```

Раз ошибка произошла на этапе авторизации, программа установила связь с сервером, но не прошла проверку комбинация «имя — пароль». Дизайнер уверен, что пароль был введён правильно. Имя, кажется, тоже набрано верно, только строчными буквами. А в карточке учётной записи первая буква имени — прописная. Дизайнер попробовал выполнить команду, записав имя пользователя с большой буквы:

```
fossil clone http://Designer@fossil.server/ sspwebsite-local.fossil
```

На этот раз команда завершилась без ошибок. Запомним на будущее, что регистр букв в имени пользователя Fossil имеет значение:

```
password for Designer: ********
remember password (Y/n)?
Round-trips: 2 Artifacts sent: 0 received: 32
Clone done, sent: 569 received: 10245 ip: 192.168.56.101
Rebuilding repository meta-data...
100.0% complete...
Extra delta compression...
Vacuuming the database...
project-id: 53e4e1445fb2a7289eb1a337ce1cf8c9a99f7b86
server-id: 5052afb115fcb681603b19f41f6c72b16b0aee2d
admin-user: Designer (password is "a83b2b")
```

Далее дизайнер создаёт рабочий каталог проекта, делает его текущим и открывает файл локального репозитория:

```
mkdir website
cd website
fossil open ..\sspwebsite-local.fossil
```

Команда выполняется без ошибок и выводит на экран следующую информацию:

```
LICENSE
index.css
index.html
project-name: Simple single-page website
repository: …/Designer/website/..\sspwebsite-local.fossil
local-root: …/Designer/website/
```

```
config-db: C:/Users/PCUser/AppData/Local/_fossil
project-code: 53e4e1445fb2a7289eb1a337ce1cf8c9a99f7b86
checkout: 307777ce6c2d287927218f08a5ecfc725ed89c01 2020-03-13
20:03:51 UTC
parent: c78dfc9a256c1bc2e02692a0fd6b2014071bae53 2020-03-07
09:56:15 UTC
tags: trunk
comment: добавлен файл лицензии MIT (user: Manager)
check-ins: 13
```

Дизайнер считает, что на этом подготовительные мероприятия завершены. Теперь надо отправить в репозиторий два файла. Один файл называется hdrbkgnd.gif и содержит фоновое изображение для заголовка. Другой файл называется demo.jpg и будет использоваться в качестве иллюстрации информационного блока.

Дизайнер копирует файлы с графическими элементами в рабочий каталог проекта и добавляет их в список файлов, управляемых системой контроля версий:

```
fossil add demo.jpg hdrbkgnd.gif
```

Fossil сообщает об успешном выполнении операции:

```
ADDED demo.jpg
ADDED hdrbkgnd.gif
```

После этого дизайнер загружает состояние рабочего каталога в репозиторий:

```
fossil commit -m "добавлены графические элементы для заголовка и
информационных блоков"
```

В ходе загрузки Fossil предупреждает, что файлы demo.jpg и hdrbkgnd.gif содержат двоичные данные, и запрашивает подтверждения на продолжение работы. Дело в том, что система контроля версий наиболее эффективно работает с текстовыми файлами, она умеет их сравнивать и находить различающиеся строки. А по поводу двоичных файлов система контроля версий может говорить только в целом, являются ли они различными или совпадают полностью. Дизайнер подтверждает продолжение работы, вводя «y» с клавиатуры:

```
Autosync: http://Designer@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 430 received: 1280 ip: 192.168.56.101
./demo.jpg contains binary data. Use --no-warnings or the "binary-
glob" setting to disable this warning.
Commit anyhow (a=all/y/N)? y
./hdrbkgnd.gif contains binary data. Use --no-warnings or the "bi-
nary-glob" setting to disable this warning.
Commit anyhow (a=all/y/N)? y
```

Запись файлов в репозиторий продолжилась, в процессе было выведено ещё несколько информационных строк, не содержащих сообщений об ошибках:

```
New_Version: 178eab2035929fecd2ee4a1a897be63323988a9970b5a7d53f3e8abe4272d6ed
Autosync: http://Designer@fossil.server:8081/
Round-trips: 1 Artifacts sent: 3 received: 0
Sync done, sent: 15526 received: 1392 ip: 192.168.56.101
```

Таким образом, дизайнер выполнил свою часть работы, и в игру снова вступает менеджер проекта. Он выполнил команду fossil update для обновления содержимого своего рабочего каталога из репозитория. Поскольку по умолчанию система Fossil находится в режиме автоматической синхронизации, то перед выгрузкой в рабочий каталог моментального снимка из последней точки сохранения, имеющейся в локальном репозитории, неявно выполнилась команда fossil pull, что привело к получению в локальный репозиторий обновлений из сетевого репозитория. Уже после этого произошла выгрузка файлов в рабочий каталог:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 12
Pull done, sent: 1251 received: 18810 ip: 192.168.56.101
ADD demo.jpg
ADD hdrbkgnd.gif
-------------------------------------------------------------------------
updated-to: 178eab2035929fecd2ee4a1a897be63323988a99 2020-03-15 16:06:32 UTC
tags: trunk
comment: добавлены графические элементы для заголовка и информаци-
онных блоков (user: Designer)
changes: 2 files modified.
"fossil undo" is available to undo changes to the working checkout.
```

Теперь в рабочем каталоге менеджера появились файлы demo.jpg и hdrbkgnd.gif, и он получил возможность ознакомиться с работой дизайнера. К самим графическим изображениям претензий не было. Однако с точки зрения структуры сайта было бы лучше разместить их в подкаталоге images. Об этом менеджер и попросил дизайнера.

Дизайнер создал в своём рабочем каталоге подкаталог images и переместил туда файлы с изображениями:

```
mkdir images
move demo.jpg images\
move hdrbkgnd.gif images\
```

В результате этих действий система контроля версий потеряла два файла из списка файлов, которыми она управляет. Чтобы убедиться в этом, можно посмотреть список произошедших в рабочем каталоге изменений с помощью команды fossil changes:

```
MISSING demo.jpg
MISSING hdrbkgnd.gif
```

В то же время система контроля версий обнаружила два бесконтрольных файла в рабочем каталоге, о чём рассказывает команда fossil extras:

```
images/demo.jpg
images/hdrbkgnd.gif
```

Устранить возникшее расхождение между системой учёта и реальным положением дел можно, выполнив в Fossil сначала команду удаления старых файлов:

```
fossil rm demo.jpg hdrbkgnd.gif
```

На экране появятся сообщения:

```
DELETED demo.jpg
DELETED hdrbkgnd.gif
```

После этого можно выполнить команду на добавление в список системы контроля версий всех файлов из каталога images:

```
fossil add images
```

Об успешном выполнении команды можно судить по сообщениям:

```
ADDED images/demo.jpg
ADDED images/hdrbkgnd.gif
```

Такого же эффекта можно было бы достичь, если после перемещения файлов в файловой системе выполнить команды перемещения их в системе контроля версий:

```
fossil mv demo.jpg hdrbkgnd.gif images\
```

Выполнение команды сопровождается выводом информационных сообщений:

```
RENAME demo.jpg images/demo.jpg
RENAME hdrbkgnd.gif images/hdrbkgnd.gif
```

Только надо помнить, что команды системы контроля версий приводят к изменениям лишь её виртуальной файловой системы. В реальной файловой системе файлы всё равно нужно перемещать с помощью команд операционной системы или файлового менеджера. Если переместить файлы только в системе контроля версий, а в файловой системе ничего не менять, то команда fossil changes сообщит о расхождении:

```
MISSING images/demo.jpg
MISSING images/hdrbkgnd.gif
```

А после того, как файлы будут перемещены в обоих системах, контроля версий и файловой, команда fossil changes сообщит лишь о произведенных изменениях:

```
RENAMED images/demo.jpg
RENAMED images/hdrbkgnd.gif
```

И всё же возможность перемещения файлов одной командой как в системе контроля версий, так и в файловой системе существует. Чтобы выполнить такую операцию, надо воспользоваться опцией *--hard*:

```
fossil mv --hard demo.jpg hdrbkgnd.gif images\
```

Команда сообщит о выполненных ею действиях:

```
RENAME demo.jpg images/demo.jpg
RENAME hdrbkgnd.gif images/hdrbkgnd.gif
MOVED_FILE …/Designer/website/demo.jpg
MOVED_FILE …/Designer/website/hdrbkgnd.gif
```

После выполнения просьбы менеджера дизайнер загрузил состояние рабочего каталога в репозиторий:

```
fossil commit -m "файлы изображений перемещены в подкаталог images"
```

Команда сообщила об успешном выполнении загрузки:

```
Autosync: http://Designer@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 427 received: 1550 ip: 192.168.56.101
New_Version:
4e6430945143ba9eaec26fb8c6ec0face75c9365e77008cd5f5ad1427f218baf
Autosync: http://Designer@fossil.server:8081/
Round-trips: 1 Artifacts sent: 1 received: 0
Sync done, sent: 2012 received: 1589 ip: 192.168.56.101
```

4.3.4. Получение отдельных файлов из репозитория

Первый верстальщик, узнав о готовности графических элементов вебсайта, захотел сразу же подключить иллюстрацию к шаблону информационного блока. Он ввёл команду fossil update для получения обновлений в свой рабочий каталог, но файлы с картинками там не появились. Хотя, судя по информационным сообщениям, команда выполнилась без ошибок:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 11
Pull done, sent: 1215 received: 18481 ip: 192.168.56.101
-------------------------------------------------------------------
checkout: f8cc457d4f7250598dee85869a8e670393e813a4 2020-03-13 20:21:18 UTC
tags: содержимое
comment: прототип блока содержимого - main, section (user: Imposer1)
changes: None. Already up-to-date
```

Почему так произошло и где находятся файлы с картинками? Посмотрим внимательно на строки, которые вывела на экран последняя команда.

Первая строка сообщает о начале автоматической синхронизации локального репозитория с сетевым. Это происходит потому, что по умолчанию в системе Fossil действует режим autosync. По этой причине команда fossil update, выполняющая обычно выгрузку мгновенного снимка, соответствующего текущей точке сохранения, из локального репозитория в рабочий каталог, перед этим провела синхронизацию репозиториев.

Во второй строке сообщается о том, что в ходе синхронизации никаких сущностей (Artifacts) не было отправлено в сетевой репозиторий (sent: 0), а из сетевого репозитория было получено одиннадцать сущностей (received: 11). По

всей видимости, среди них есть и интересующие верстальщика файлы с изображениями.

Третья строка сообщает о том, что получение информации из сетевого репозитория завершено (Pull done), и отчитывается о деталях сетевого обмена: сколько байт было отправлено (sent: 1215) и сколько было получено (received: 18481).

Интересно, что, несмотря на очевидный факт получения информации из сетевого репозитория, где уже точно есть графические файлы, в последней строке сообщается о том, что рабочий каталог находится в актуальном состоянии, и для него нет никаких изменений (changes: None. Already up-to-date).

А вот теперь мы вплотную подошли к выяснению причины неполучения требуемых файлов. В строке checkout: указан идентификатор точки сохранения, из которой мгновенный снимок выгружается в рабочий каталог, а в строке tags: указано название ветки репозитория, на которой эта точка сохранения находится. Сравним полученные значения с теми, что были выведены дизайнеру, когда он делал выгрузку из репозитория в свой рабочий каталог:

```
…
checkout: 307777ce6c2d287927218f08a5ecfc725ed89c01 2020-03-13
20:03:51 UTC
parent: c78dfc9a256c1bc2e02692a0fd6b2014071bae53 2020-03-07
09:56:15 UTC
tags: trunk
comment: добавлен файл лицензии MIT (user: Manager)
```

Дизайнер не выбирал ветвь репозитория, в которой ему предстояло работать, поэтому автоматически ему была назначена основная ветвь разработки — trunk. В эту ветвь он и загрузил свои файлы с графическими элементами. Менеджер, в отличие от верстальщиков, тоже не переключался с основной ветви разработки, поэтому у него не возникло проблем с получением файлов от дизайнера. А вот верстальщик ведёт работу в отдельной ветке «содержимое», для которой в репозитории действительно нет никаких новостей.

С причиной сложившейся ситуации разобрались. Но как же верстальщику всё-таки получить файл с иллюстрацией информационного блока?

Первая мысль, которая пришла в голову верстальщику Imposer1, была: «Надо создать ещё один рабочий каталог (Fossil позволяет использовать множество рабочих каталогов к одному репозиторию), выгрузить в него моментальный снимок из точки сохранения, которую создал дизайнер и которая содержит файлы с изображениями. После этого — переписать из него требуемый файл в основной рабочий каталог и продолжать работу».

Такой вариант действительно имеет право на существование. Но нельзя ли всё сделать проще, без лишних сущностей в виде вспомогательных рабочих каталогов? Верстальщик не стал торопиться, а решил сначала осмотреться.

Чтобы узнать, в моментальном снимке какой точки сохранения находится требуемый файл, он выполнил команду:

```
fossil timeline -t ci -n 2
```

Репозиторий Fossil используется для хранения всех сущностей, которые имеются в этой системе: файлов, сообщений форума, статей документации, заявок на доработку. Чтобы отображались только точки сохранения, соответствующие операциям над моментальными снимками рабочих каталогов, в команде использован ключ *-t* со значением ci. Ключ *-n* со значением 2 ограничивает выдачу двумя последними записями:

```
=== 2020-03-15 === 18:46:48 [4e64309451] файлы изображений перемещены
в подкаталог images (user: Designer tags: trunk)
16:06:32 [178eab2035] добавлены графические элементы для заголовка
и информационных блоков (user: Designer tags: trunk)
--- entry limit (2) reached ---
```

Из описаний становится понятно, что интересующая точка сохранения имеет идентификатор 4e64309451. Верстальщик Imposer1 решил посмотреть, как выглядят файлы в мгновенном снимке, соответствующем этой точке сохранения, для чего воспользовался командой:

```
fossil ls -r 4e64309451
```

Команда вывела список файлов, в котором присутствовали и загруженные в репозиторий дизайнером:

```
images/demo.jpg
images/hdrbkgnd.gif
index.css
index.html
LICENSE
```

Отлично, файл с иллюстрацией информационного блока обнаружен, это images/demo.jpg. Осталось выгрузить его из мгновенного снимка в рабочий каталог. Для этого верстальщик ввёл команду:

```
fossil update 4e64309451 images/demo.jpg
```

Она выполнилась без ошибок и вывела на экран следующую информацию:

```
Autosync: http://Imposer1@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 398 received: 1628 ip: 192.168.56.101
ADD images/demo.jpg ДОБАВЛЕН images/demo.jpg
-------------------------------------------------------------------------------
updated-to: 4e6430945143ba9eaec26fb8c6ec0face75c9365 2020-03-15 18:46:48 UTC
tags: trunk
comment: файлы изображений перемещены в подкаталог images (user: Designer)
changes: 1 file modified. изменения: 1 файл модифицирован.
"fossil undo" is available to undo changes to the working checkout.
```

В результате выполнения последней команды в рабочем каталоге первого верстальщика был создан подкаталог images, а в него записан файл demo.jpg. Цель достигнута, верстальщик может продолжать работу, используя изображение с иллюстрацией для информационного блока.

Отметим, что в остальном состояние рабочего каталога не изменилось. Он по-прежнему базируется на мгновенном снимке из точки сохранения, находящейся на ветке содержимое репозитория. Это подтверждается выдачей команды fossil status:

```
repository: …/Imposer1/website/..\sspwebsite-local.fossil
local-root: …/Imposer1/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: f8cc457d4f7250598dee85869a8e670393e813a4 2020-03-13 20:21:18 UTC
parent: ddc308408924cddce8a39c2f79f51270bad44394 2020-03-13 18:00:58 UTC
tags: содержимое
comment: прототип блока содержимого - main, section (user: Imposer1)
```

И вообще, состояние репозитория, как сетевого, так и локального, не изменилось. Получение файла никак не отразилось на временной шкале (рис. 4.20).

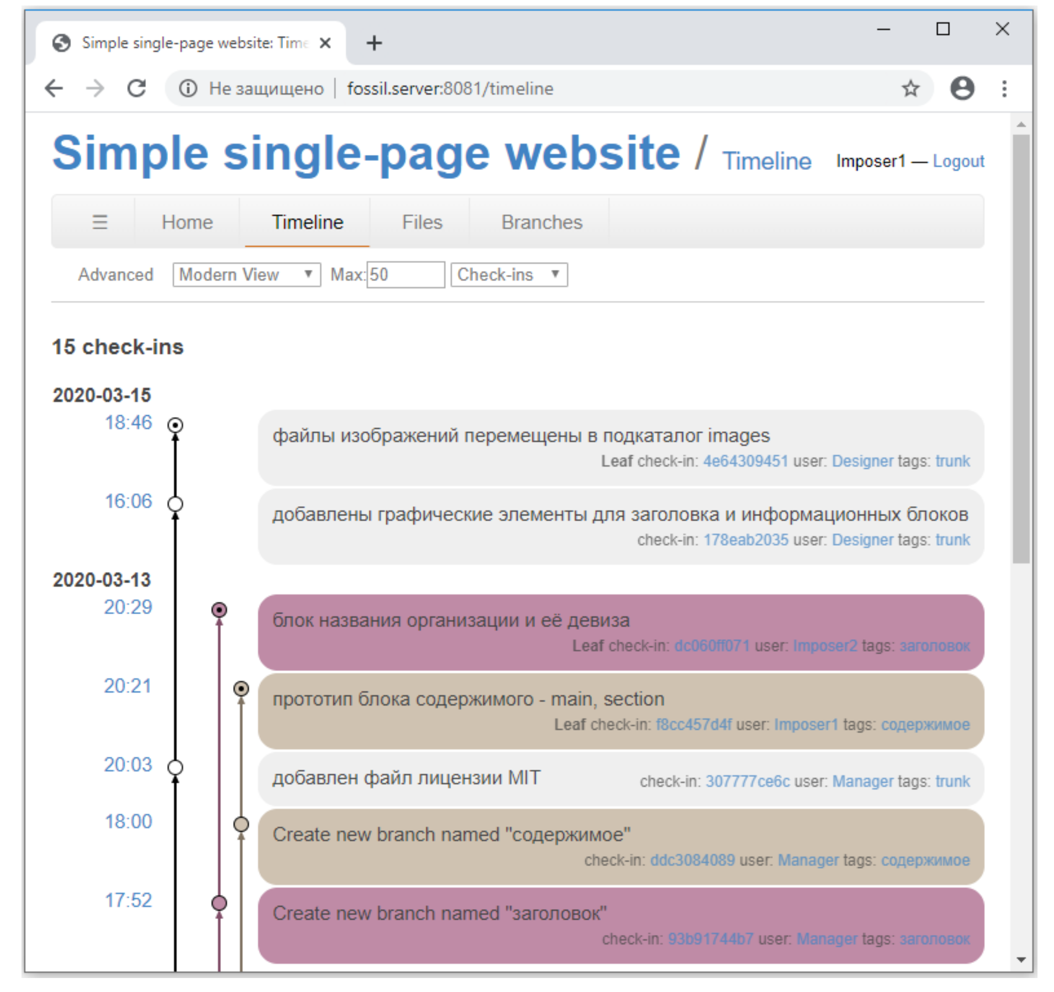


Рис. 4.20

Получение отдельных файлов из моментальных снимков в рабочий каталог не отражается в истории репозитория

Более того, над выгруженным из репозитория файлом отсутствует контроль со стороны системы контроля версий, о чём сообщает команда fossil extras:

```
images/demo.jpg
```

Так может быть, стоит добавить этот файл в список контроля с помощью команды fossil add? Но зачем? Верстальщик Imposer1 не собирается его дорабатывать, этим занимается дизайнер. Так пусть он и сопровождает его в основной ветке проекта — trunk. А первый верстальщик может продолжать работу над содержимым веб-страницы, модифицируя HTML- и CSS-файлы в своей ветке «содержимое» и используя полученный файл images/demo.jpg в качестве вспомогательного элемента.

Позже, когда верстальщик выполнит свою работу, менеджер проекта выполнит слияние ветки «содержимое» с веткой trunk, и проект обретёт целостность. А пока верстальщику нет необходимости контролировать изменения в файле demo.jpg.

У читателя может ещё возникнуть вопрос, откуда верстальщик Imposer1 узнал о таких полезных ключах и параметрах команд Fossil, которые позволили ему получить требуемый файл без выгрузки мгновенного снимка в дополнительный рабочий каталог? Дело в том, что Imposer1 владеет английским языком в пределах, достаточных для чтения технической документации, а справку по любой команде можно получить прямо от Fossil, например так:

```
fossil help ls
```

Или так:

```
fossil update --help
```

Перечень всех команд Fossil можно вывести на экран с помощью команды:

```
fossil help
```

4.3.5. Слияние ветвей

Второй верстальщик проконсультировался у первого верстальщика относительно способа получения файла с изображением из основной ветки репозитория в свой рабочий каталог и повторил его действия.

Сначала он запросил выгрузку изменений из репозитория в рабочий каталог, которая, в свою очередь, привела к синхронизации локального репозитория с сетевым:

```
fossil update
```

Затем он узнал идентификатор точки сохранения, моментальный снимок которой содержит требуемый файл:

```
fossil timeline -t ci -n 2
```

Потом посмотрел, какие файлы имеются в моментальном снимке:

```
fossil ls -r 4e64309451
```

И, наконец, выгрузил из моментального снимка, хранящегося в репозитории, в свой рабочий каталог файл images/hdrbkgnd.gif:

```
fossil update 4e64309451 images/hdrbkgnd.gif
```

После этого оба верстальщика продолжили работу над своими задачами. Они продвигались к цели небольшими шагами, завершая каждый этап созданием точки сохранения в репозитории с помощью команды fossil commit. Через непродолжительное время верстальщики пришли к выводу, что готовы предоставить результаты работы в основную ветку проекта (рис. 4.21 и 4.22).

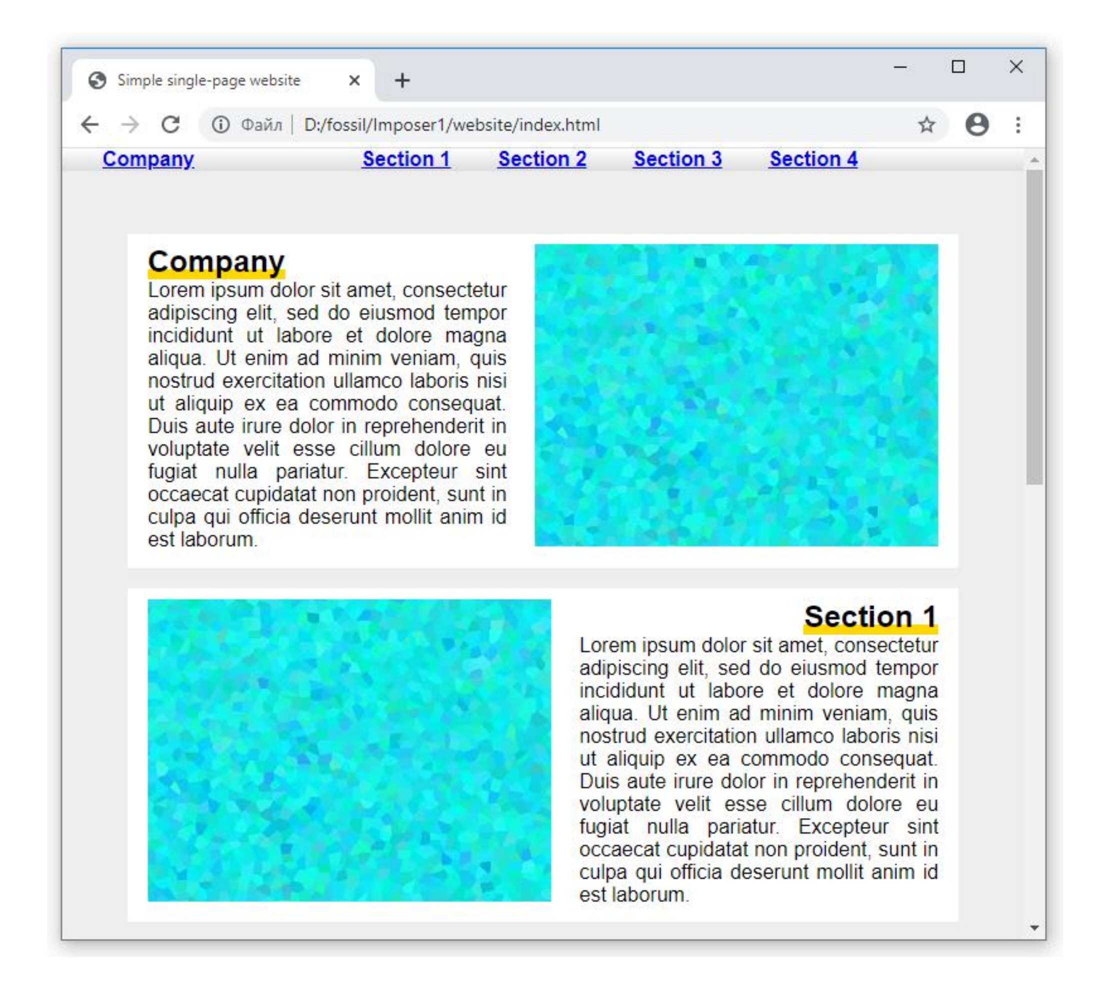


Рис. 4.21

*Результат работы первого верстальщика —
содержимое веб-странички*

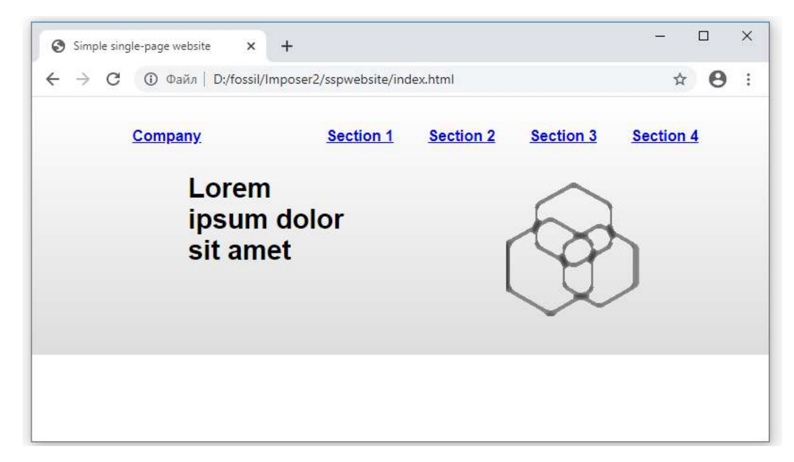


Рис. 4.22

Результат работы второго верстальщика — заголовок веб-странички

Непосредственно перед слиянием шкала времени проекта выглядит так, как показано на рисунке 4.23.

Рис. 4.23

Шкала времени перед слиянием ветвей

Слиянием ветвей менеджер решил заняться самостоятельно. Прежде всего он, уже по привычке, выполнил команду выгрузки последних обновлений из репозитория в свой рабочий каталог:

```
fossil update
```

В результае на его рабочем каталоге отразилось перемещение дизайнером файлов изображений в созданный для них подкаталог images, а также был получен ещё один файл h2bkgnd.gif, содержащий фоновое изображение для заголовков информационных блоков:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 2 Artifacts sent: 0 received: 29
Pull done, sent: 1906 received: 8689 ip: 192.168.56.101
REMOVE demo.jpg
REMOVE hdrbkgnd.gif
ADD images/demo.jpg
ADD images/h2bkgnd.gif
ADD images/hdrbkgnd.gif
-----------------------------------------------------------------
updated-to: 2672cdc02b453f65bae76ed9bad7559895f50a92 2020-03-17 09:41:18 UTC
tags: trunk
comment: добавлено фоновое изображение для заголовков информационных блоков (user: Designer)
changes: 5 files modified.
"fossil undo" is available to undo changes to the working checkout.
```

Исходные тексты, находящиеся в рабочем каталоге менеджера, формируют очень невзрачную веб-страничку (рис. 4.24):

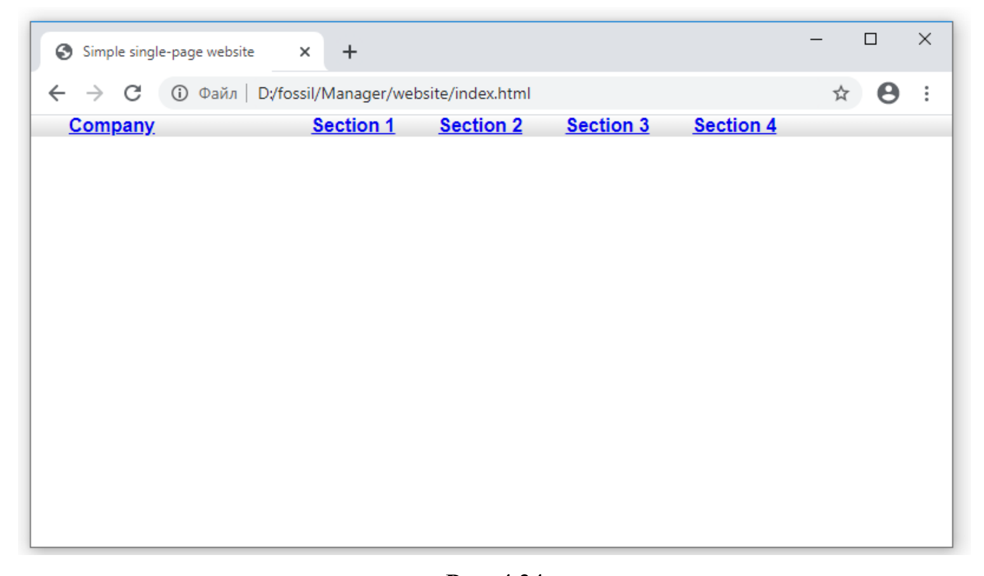


Рис. 4.24

Веб-страничка по исходным текстам основной ветки репозитория

Менеджер решил напомнить себе, какие ветви проекта развиваются сейчас в репозитории:

```
fossil branch
```

Команда сообщила о трёх активных ветвях, звёздочка отмечала, что рабочий каталог менеджера отражает состояние основной ветви репозитория trunk:

```
* trunk
заголовок
содержимое
```

Начать слияние менеджер решил с заголовка. Чтобы узнать идентификатор последней точки сохранения ветви «заголовок», менеджер ввёл команду:

```
fossil branch info "заголовок"
```

Из ответа команды выяснилось, что искомый идентификатор равен 4aec8726e4b2b1f4:

```
заголовок: open as of 2020-03-17 09:33:13 on 4aec8726e4b2b1f4
```

Для того, чтобы влить ветвь «заголовок» в точку сохранения, которой соответствует рабочий каталог, менеджер выполнил команду:

```
fossil merge 4aec8726e4b2b1f4
```

Команда выполнилась без проблем, сообщив о том, что в ходе её выполнения были обновлены два файла:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 395 received: 2678 ip: 192.168.56.101
UPDATE index.css
UPDATE index.html
"fossil undo" is available to undo changes to the working checkout.
```

Для уточнения состояния рабочего каталога менеджер выполнил команду:

```
fossil status
```

Информация, выданная командой, в подробностях описывает выполненное вливание ветви «заголовок» в основную ветвь репозитория.

```
repository: …/Manager/website/..\sspwebsite-local.fossil
local-root: …/Manager/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: 2672cdc02b453f65bae76ed9bad7559895f50a92 2020-03-17 09:41:18 UTC
parent: 4e6430945143ba9eaec26fb8c6ec0face75c9365 2020-03-15 18:46:48 UTC
tags: trunk
comment: добавлено фоновое изображение для заголовков информационных блоков (user: Designer)
UPDATED_BY_MERGE index.css
UPDATED_BY_MERGE index.html
MERGED_WITH
4aec8726e4b2b1f43bfe2f61401ca2a56a22ce71b0b470767b6cc6954ddc29ab
```

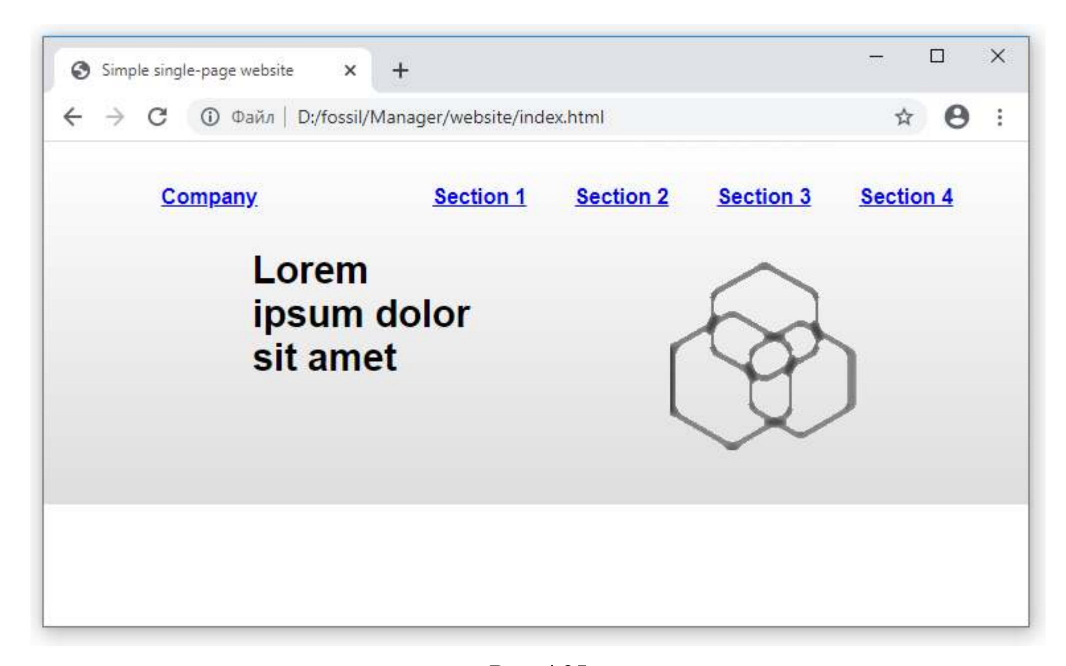


Рис. 4.25

Веб-страничка после получения заголовка

Слияние произошло бесконфликтно. Система Fossil самостоятельно справилась с анализом тех незначительных изменений, которые внёс в исходные файлы проекта верстальщик Imposer2. Сам факт слияния пока что никак не отразился на репозитории, он произошёл только в рабочем каталоге менеджера. Если бы результат слияния менеджера не устроил, он мог бы выполнить команду fossil undo, и рабочий каталог был бы возвращён в предыдущее состояние. Однако, ознакомившись с результатом (рис. 4.25), он решил зафиксировать его в репозитории с помощью команды:

```
fossil commit -m "принят заголовок веб-страницы"
```

Как обычно, команда отчиталась о результате своей работы:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 430 received: 2677 ip: 192.168.56.101
New_Version:
56e4a0c8f6cc7de64df1c2a2ecf2fad4c470f208d8821b7ed2359022e749762e
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 1 received: 0
Sync done, sent: 3084 received: 2716 ip: 192.168.56.101
```

После загрузки моментального снимка рабочего каталога менеджера в репозиторий факт слияния ветви «заголовок» с основной линией разработки нашёл отражение в репозитории, и шкала времени проекта приобрела вид, показанный на рисунке 4.26.

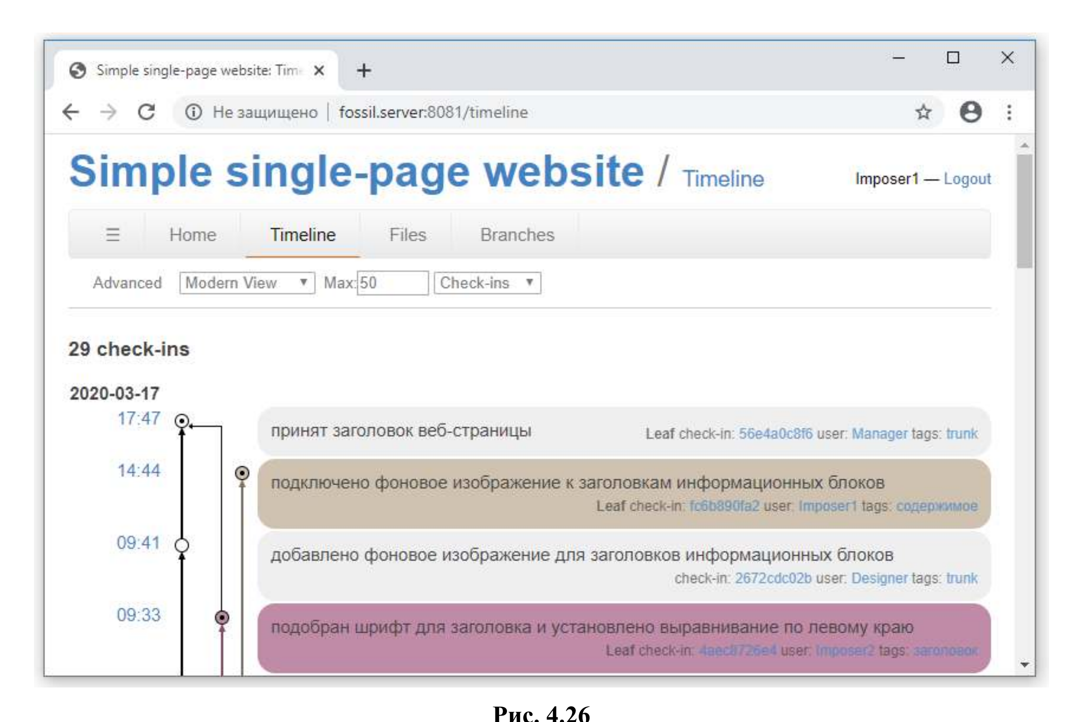


Рис. 4.26

Шкала времени после получения заголовка

Менеджер ещё раз открыл в веб-браузере файл index.html из своего рабочего каталога и внимательно рассмотрел заголовок веб-страницы. В целом всё выглядело неплохо, но гиперссылки, образующие горизонтальное меню, со своим ярким синим цветом и традиционным подчёркиванием выглядели слишком аляповато.

Поскольку рабочий день только что закончился и верстальщика уже не было на рабочем месте, менеджер самостоятельно подправил стили в файле index.css. Результат работы он записал в репозиторий командой:

```
fossil commit -m "исправлен шрифт горизонтального меню в заголовке странички"
```

Теперь пришло время заняться содержимым веб-странички. Менеджер запросил информацию о ветке репозитория, которую развивал верстальщик Imposer1, командой:

```
fossil branch info "содержимое"
```

Эта команда вывела на экран строку:

```
содержимое: open as of 2020-03-17 14:44:46 on fc6b890fa2454143
```

Из полученной информации стало понятно, что идентификатор последней точки сохранения на ветви «содержимое» равен fc6b890fa2454143. Для слияния ветви «содержимое» с основной ветвью проекта менеджер ввёл команду:

```
fossil merge fc6b890fa2454143
```

В ходе слияния система Fossil не смогла автоматически наложить изменения из вливаемой ветви на файл index.css в рабочем каталоге менеджера и сообщила о конфликте:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 398 received: 2793 ip: 192.168.56.101
MERGE index.css СЛИЯНИЕ index.css
***** 1 merge conflicts in index.css 1 конфликт слияния в index.css
MERGE index.html СЛИЯНИЕ index.html
WARNING: 1 merge conflicts ПРЕДУПРЕЖДЕНИЕ: 1 конфликт слияния
"fossil undo" is available to undo changes to the working checkout.
```

Менеджер уже второй раз на этом проекте столкнулся с конфликтом слияния, поэтому не стал паниковать, тем более что выполненная операция пока что произвела изменения только в его рабочем каталоге, а не в репозитории, и при необходимости её можно отменить командой fossil undo. Он решил ознакомиться с возникшей ситуацией более подробно и в первую очередь выполнил команду:

```
fossil status
```

Как обычно, команда вывела на экран несколько строк с информацией:

```
repository: …/Manager/website/..\sspwebsite-local.fossil
local-root: …/Manager/website/
config-db: C:/Users/PCUser/AppData/Local/_fossil
checkout: a2d5807ce2f2b2bf4d1069dbc3deff5ad13e36dd 2020-03-18 06:42:20 UTC
parent: 56e4a0c8f6cc7de64df1c2a2ecf2fad4c470f208 2020-03-17 17:47:37 UTC
tags: trunk
comment: исправлен шрифт горизонтального меню в заголовке странич-
ки (user: Manager)
CONFLICT index.css
EDITED index.html
MERGED_WITH
fc6b890fa24541434d863d3faf3e9f5a3fee0e5cac6a6c821c132c4ad994feb5
```

Последняя строка говорит о том, что действие, которое было выполнено над рабочим каталогом, — это слияние с ветвью fc6b890fa… Предпоследняя строка информирует, что в результате слияния файл index.html был изменён. А строка, ей предшествующая, сообщает о том, что при попытке наложения изменений на файл index.css были обнаружены перекрывающиеся участки, и это вылилось в конфликт и необходимость ручного вмешательства.

Как всегда, при конфликте слияния система Fossil создала в рабочем каталоге три варианта проблемного файла:

- index.css-baseline — файл из моментального снимка репозитория, который соответствует той точке сохранения, в которой было выполнено ответвление (т. е. последней общей точке сохранения для обеих ветвей, участвующих в слиянии);
- index.css-merge — файл из вливаемой ветви репозитория;

- index.css-original — резервная копия файла index.css из рабочего каталога на момент, предшествующий слиянию.

Листинг. index.css-baseline

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header {background: gray; background: linear-gradient(to top,#DDD,#FFF)}
header nav {font-weight: bold; font-decoration: undercore}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; dis-
play: inline}
header nav ul li:first-child {margin-right: 6em}
```

Листинг. index.css-merge

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header {background: gray; background: linear-gradient(to top,#DDD,#FFF)}
header nav {font-weight: bold; font-decoration: undercore}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; dis-
play: inline}
header nav ul li:first-child {margin-right: 6em}
main {max-width: 1000px; margin: auto;  padding: 4ex 2%; back-
ground-color: #EEE}
section {margin-top: 2ex; padding: 1ex 1em; text-align: justify}
section {margin-left: 5%; margin-right: 5%}
section {min-height: 200px; background: white}
section header {background: none}
section header h2 {background-image: url(images/h2bkgnd.gif);
   background-position: bottom; background-repeat: repeat-x}
section img {margin-bottom: 1ex}
section:nth-child(odd) header {text-align: left; float: left}
section:nth-child(odd) img {float: right; margin-left: 1.4em}
section:nth-child(odd) div {clear: left}
section:nth-child(even) header {text-align: right; float: right}
section:nth-child(even) img {float: left; margin-right: 1.4em}
section:nth-child(even) div {clear: right}
section hr {clear: both; border: 0px solid}
```

Листинг. index.css-original

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header {max-width: 1000px; margin: auto; padding: 4ex 2%; text-align: center}
header {background: gray; background: linear-gradient(to top,#DDD,#FFF)}
header nav {font-weight: bold; font-decoration: none}
header nav ul li a {color: #777; text-decoration: none}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; display: inline}
header nav ul li:first-child {margin-right: 6em}
header div {height: 160px; margin-top: 2ex; padding-left: 20%}
header div {text-align: left; font: normal normal bold 1.8em sans-serif}
```

```
header div {background-image: url(images/hdrbkgnd.gif);
   background-position: 80% center; background-repeat: no-repeat}
```

Листинг. index.css непосредственно после слияния

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header {max-width: 1000px; margin: auto; padding: 4ex 2%; text-align: center}
header {background: gray; background: linear-gradient(to top,#DDD,#FFF)}
header nav {font-weight: bold; font-decoration: none}
header nav ul li a {color: #777; text-decoration: none}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; display: inline}
header nav ul li:first-child {margin-right: 6em}
<<<<<<< BEGIN MERGE CONFLICT: local copy shown first <<<<<<<<<<<<<
header div {height: 160px; margin-top: 2ex; padding-left: 20%}
header div {text-align: left; font: normal normal bold 1.8em sans-
serif}
header div {background-image: url(images/hdrbkgnd.gif);
   background-position: 80% center; background-repeat: no-repeat}
======= COMMON ANCESTOR content follows ============================
======= MERGED IN content follows ==================================
main {max-width: 1000px; margin: auto;  padding: 4ex 2%; back-
ground-color: #EEE}
section {margin-top: 2ex; padding: 1ex 1em; text-align: justify}
section {margin-left: 5%; margin-right: 5%}
section {min-height: 200px; background: white}
section header {background: none}
section header h2 {background-image: url(images/h2bkgnd.gif);
   background-position: bottom; background-repeat: repeat-x}
section img {margin-bottom: 1ex}
section:nth-child(odd) header {text-align: left; float: left}
section:nth-child(odd) img {float: right; margin-left: 1.4em}
section:nth-child(odd) div {clear: left}
section:nth-child(even) header {text-align: right; float: right}
section:nth-child(even) img {float: left; margin-right: 1.4em}
section:nth-child(even) div {clear: right}
section hr {clear: both; border: 0px solid}
>>>>>>> END MERGE CONFLICT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

При выполнении слияния двух вариантов файла index.css (существующего и вливаемого) система Fossil определила часть этого файла, общую для обоих вариантов, и отделила её от остальной части строкой < BEGIN MERGE CONFLICT: local copy shown first < (НАЧАЛО КОНФЛИКТА СЛИЯНИЯ: сначала показана локальная копия). После неё система Fossil разместила строки, имеющиеся в варианте файла рабочего каталога, но отсутствующие во вливаемом варианте, и завершила этот блок строкой = COMMON ANCESTOR content follows = (далее идёт содержимое ОБЩЕГО ПРЕДКА).

По идее, после этой строки должен идти текст, который присутствовал в файле до создания ветви «содержимое», но потом был изменён как в основной ветви, так и в её ответвлении. В данном случае такой текст отсутствует, поэтому соответствующий ему блок пуст.

Далее, после строки = MERGED IN content follows = (содержимое ВЛИВАЕМОГО ФАЙЛА), следует текст, имеющийся в варианте файла index.css из ветви «содержимое» и перекрывающий соответствующий текст в варианте файла рабочего каталога. Строка > END MERGE CONFLICT > (КОНЕЦ КОНФЛИКТА СЛИЯНИЯ) завершает блок различающихся участков файла index.css.

В данном примере решение конфликта не составляет трудности. Поскольку с момента ответвления оба верстальщика записывали в файл index.css правила стилевого оформления для той части веб-странички, над которой они работали, то оба конфликтующих с точки зрения Fossil блока должны быть сохранены в результирующем файле index.css.

Поэтому менеджер просто удалил из файла index.css служебные строки, вставленные системой Fossil для разделения блоков, и открыл файл index.html в веб-браузере, чтобы убедиться в правильности своих предположений (рис. 4.27).

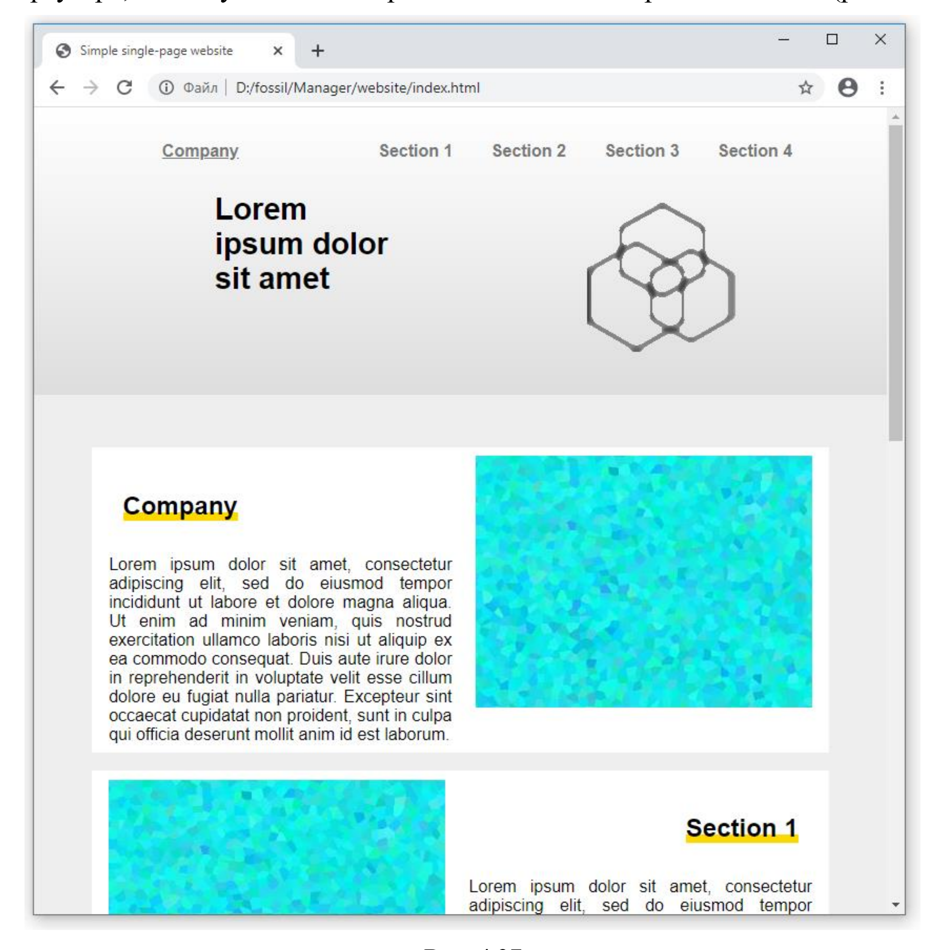


Рис. 4.27

Веб-страница после вливания ветви «содержимое»

Это был именно тот результат, что и требовался. Перед записью моментального снимка рабочего каталога в репозиторий менеджер ещё раз проверил

состояние с помощью команды fossil changes — файл index.css больше не был конфликтующим, а стал исправленным, как и файл index.html:

```
EDITED index.css
EDITED index.html
MERGED_WITH
fc6b890fa24541434d863d3faf3e9f5a3fee0e5cac6a6c821c132c4ad994feb5
```

Менеджер создал точку сохранения в репозитории командой:

```
fossil commit -m "принято содержимое веб-страницы"
```

Из-за того, что он в своё время не выполнил дополнительную настройку Fossil, ему пришлось давать разрешения на запись в репозиторий текстовых файлов, у которых в качестве символов перевода строк используются двухбайтовые комбинации CR/LF:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 432 received: 2793 ip: 192.168.56.101
./index.css contains CR/LF line endings. Use --no-warnings or the
"crlf-glob" setting to disable this warning.
Commit anyhow (a=all/c=convert/y/N)? y
./index.html contains CR/LF line endings. Use --no-warnings or the
"crlf-glob" setting to disable this warning.
Commit anyhow (a=all/c=convert/y/N)? y
New_Version:
9b1e92e400c3f2b6f1553a9ffb44c2e452959d19fcca5bd2cd2728a817ee2fab
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 3 received: 0
Sync done, sent: 4302 received: 2905 ip: 192.168.56.101
```

После вливания ветви «содержимое» в основную ветвь репозитория шкала времени проекта приобрела вид, показанный на рисунке 4.28.

Казалось бы, на этом можно и остановиться. Но если выполнить команду fossil extras, то она выведет на экран список забытых файлов, автоматически созданных Fossil в помощь для решения конфликта слияния:

```
index.css-baseline
index.css-merge
index.css-original
```

На этот раз файлы не пригодились, и их можно удалить из рабочего каталога командой del *-* (в Windows) или rm *-* (в UNIX и Linux).

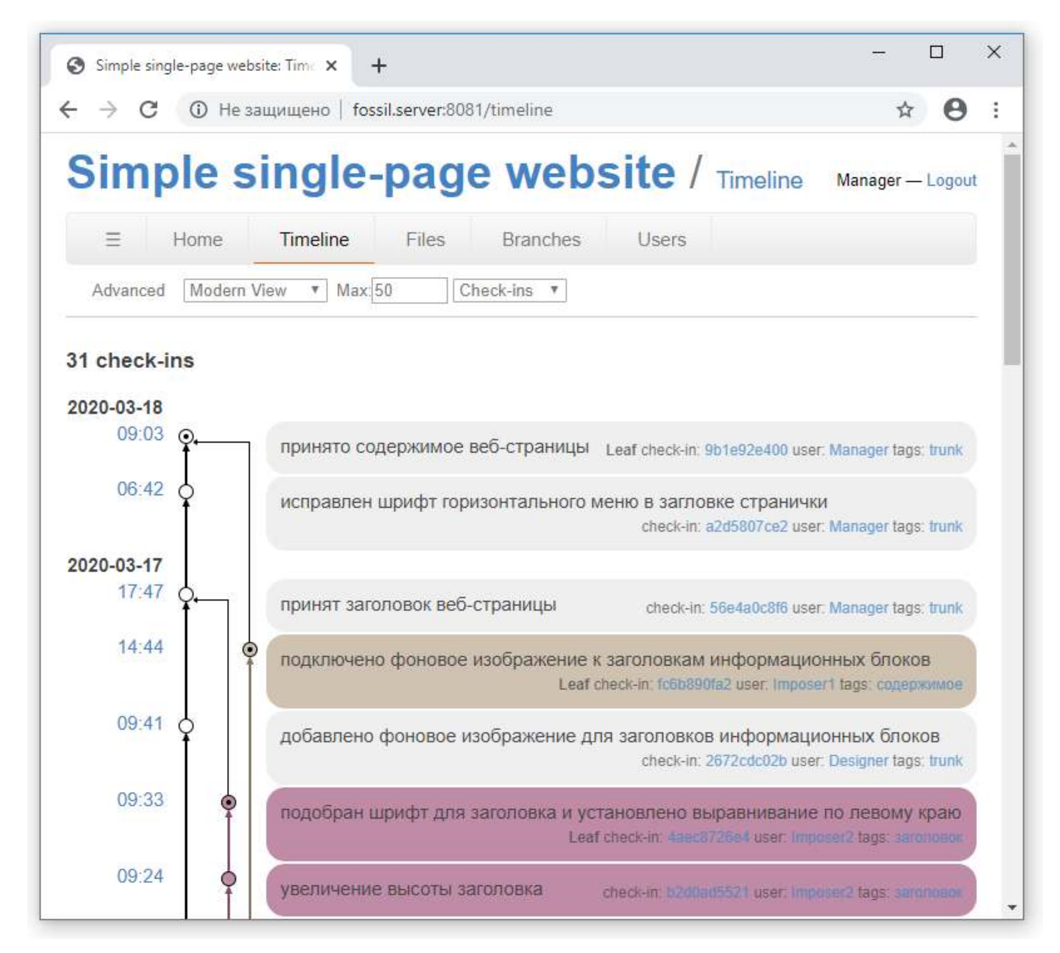


Рис. 4.28

Шкала времени проекта после вливания ветви «содержимое»

4.4. Ярлыки и свойства

4.4.1. Пользовательские идентификаторы

При работе над проектом средствами Fossil довольно часто приходится использовать идентификаторы, чтобы указать командам, над какими именно точками сохранения требуется выполнить ту или иную операцию. Эти идентификаторы, автоматически присваиваемые системой Fossil всем появляющимся в ней объектам, представляют собой длинные случайные последовательности шестнадцатеричных цифр, которые не только сложно запомнить, но непросто даже ввести с клавиатуры без ошибок. В связи с этим возникает мысль о том, что было бы неплохо некоторым точкам сохранения назначить более удобные идентификаторы.

И система Fossil такую возможность предоставляет. Каждая точка сохранения в репозитории может снабжаться ярлыком (tag) — коротким текстом. Яр-

лыки бывают точечными и распространяющимися. Точечные ярлыки «приклеиваются» к точке сохранения и относятся только к ней. Распространяющиеся ярлыки «приклеиваются» не только к указанной точке сохранения, но и ко всем следующим после неё точкам, находящимся на этой ветви. Кроме того, при создании очередных точек сохранения в той же ветви такие ярлыки автоматически копируются и на них.

Когда создаётся новая ветвь в репозитории, её первой точке сохранения автоматически назначается распространяющийся ярлык с таким же именем, какое имеет созданная ветвь. Посмотреть список всех использующихся в репозитории ярлыков можно с помощью команды:

```
fossil tag list
```

Поскольку до настоящего момента никаких ярлыков специально не создавалось, она выведет на экран список наименований имеющихся в репозитории ветвей:

```
trunk
заголовок
содержимое
```

При необходимости можно вывести список сущностей, хранящихся в репозитории и имеющих определённый ярлык. Например, вывести список точек сохранения на ветви «заголовок» позволяет следующая команда:

```
fossil tag find -t ci "заголовок"
```

В этой команде значение параметра *-t*, равное ci (check-in), указывает, что ожидается вывод только точек сохранения с отфильтровыванием других хранящихся в базе данных элементов. В конце команды указывается значение ярлыка, которое интересует пользователя. Для рассматриваемого примера команда выведет на экран:

```
=== 2020-03-17 ===
09:33:13 [4aec8726e4] подобран шрифт для заголовка и установлено выравнивание
по левому краю (user: Imposer2 tags: заголовок)
09:24:09 [b2d0ad5521] увеличение высоты заголовка (user: Imposer2 tags:
заголовок)
06:49:50 [dbaf60e2bc] подключено фоновое изображение для заголовка (user:
Imposer2 tags: заголовок)
=== 2020-03-16 ===
18:55:29 [7b14ea86a2] центрирование содержимого заголовка (user: Imposer2
tags: заголовок)
18:52:17 [3d6f997e13] центрирование блока заголовка (user: Imposer2 tags:
заголовок)
=== 2020-03-13 ===
20:29:14 [dc060ff071] блок названия организации и её девиза (user: Imposer2
tags: заголовок)
17:52:44 [93b91744b7] Create new branch named "заголовок" (user: Manager tags:
заголовок)
+++ no more data (7) +++
```

Название ярлыка можно использовать вместо идентификатора. Выше был рассмотрен пример, в котором верстальщики выгружали из репозитория в свои рабочие каталоги файлы с изображениями, находящиеся в мгновенном снимке рабочего каталога дизайнера. Для доступа к этому мгновенному снимку использовался идентификатор точки сохранения.

Возможно, было бы удобнее назначить этой точке сохранения ярлык и обращаться к ней по имени. Попробуем это сделать.

4.4.2. Точечные ярлыки

Для «навешивания» ярлыка введём команду:

```
fossil tag add "изображения" 4e64309451
```

В этой команде «изображения» — это название ярлыка, а 4e64309451 — идентификатор точки сохранения. Если допустить ошибку в идентификаторе, то будет выведено сообщение not found: (не найден) и указан набранный идентификатор. Если всё в порядке, то команда ничего не сообщит.

Теперь для того, чтобы посмотреть список файлов, находящихся в мгновенном снимке точки сохранения с идентификатором 4e64309451, можно набрать команду:

```
fossil ls -r "изображения"
```

Если ярлык точке сохранения назначил менеджер и он же выполнил команду просмотра, то на экран будет выведен список файлов соответствующего мгновенного снимка. Если же команду просмотра выполнит другой участник проекта, например первый верстальщик, то ему будет выдано сообщение об ошибке: not a valid check-in: (недопустимая точка сохранения:) изображения.

Это не удивительно, потому что информация о назначении ярлыка ещё не попала в его репозиторий. Чтобы другие участники проекта могли пользоваться ярлыком, менеджер должен сначала загрузить его в репозиторий, а те, в свою очередь, должны выполнить синхронизацию своих репозиториев.

Ситуация несколько усложняется тем обстоятельством, что сам факт назначения ярлыка система Fossil не считает событием, изменяющим состояние рабочего каталога. Оно не отражается в выводе команды fossil status, а попытка загрузки в репозиторий с помощью команды fossil commit завершается сообщением: nothing has changed; use --allow-empty to override (ничего не изменилось; используйте --allow-empty для переопределения).

Конечно, ярлык будет загружен в репозиторий в дальнейшем при загрузке изменённых файлов рабочего каталога. Но у менеджера необходимость редактирования файлов может возникнуть нескоро. К счастью, он может воспользоваться подсказкой Fossil и указать опцию *--allow-empty*:

```
fossil commit --allow-empty -m "назначен ярлык \"изображения\"
точке сохранения с графическими элементами"
```

После этого другие участники проекта могут выполнить команду fossil update, чтобы получить информацию о назначенном ярлыке в свои репозитории. Несмотря на то, что команда после выполнения сетевого обмена выведет сооб-

щение changes: None. Already up-to-date (изменения: Ничего. Уже актуально), на самом деле ярлык будет получен.

В этом можно убедиться, просмотрев список ярлыков командой fossil tag list или выполнив приведенную выше команду просмотра списка файлов в мгновенном снимке, — она действительно отобразит имена файлов, а не сообщение об ошибке. Увидеть результат назначения ярлыка точке сохранения можно и на шкале времени репозитория (рис. 4.29).

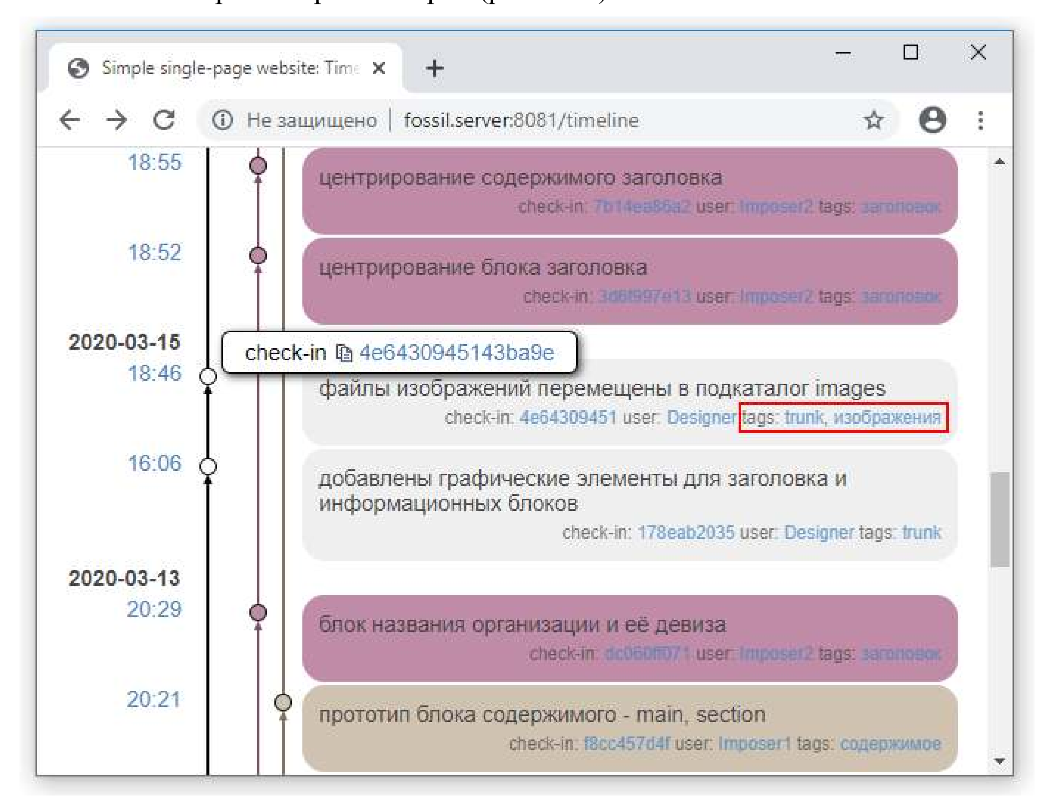


Рис. 4.29

Точка сохранения с идентификатором 4e64309451 получила ярлык изображения наряду с распространяющимся по ветке ярлыком trunk

Однако ярлыки нельзя рассматривать как полную альтернативу идентификаторам точек сохранения. На ярлыки не распространяется требование уникальности, и наиболее яркий тому пример — названия ветвей разработки. Все точки сохранения, находящиеся на одной ветви разработки, имеют ярлыки с одинаковыми названиями — trunk, заголовок, содержимое. Можно и явным образом «навесить» одноимённые ярлыки на несколько различных точек сохранения. Например, дизайнер отдельно загрузил в репозиторий фоновое изображение для заголовков информационных блоков. Этой точке сохранения тоже можно назначить ярлык изображения:

```
fossil tag add "изображения" 2672cdc02b
```

Теперь в рабочем каталоге менеджера две точки сохранения имеют ярлыки с одним и тем же названием, что демонстрирует команда:

```
fossil tag find "изображения"
=== 2020-03-17 ===
09:41:18 [2672cdc02b] добавлено фоновое изображение для заголовков
информационных блоков (user: Designer tags: trunk, изображения)
=== 2020-03-15 ===
18:46:48 [4e64309451] файлы изображений перемещены в подкаталог
images (user:
Designer tags: trunk, изображения)
+++ no more data (2) +++
```

Почему сказано «в рабочем каталоге менеджера», а не «в репозитории»? Потому что назначенные менеджером ярлыки попадут в репозиторий только после очередной загрузки моментального снимка рабочего каталога.

В веб-интерфейсе Fossil список действующих в репозитории нераспространяющихся ярлыков, подобных только что созданному, можно посмотреть либо через пункт горизонтального меню Tags, либо через раскрывающееся меню Branches-Tags, либо по URL: http://fossil.server:8081/taglist (если сетевой репозиторий находится на сервере fossil.server и ожидает подключений на TCP-порту 8081), как проиллюстрировано рисунком 4.30.

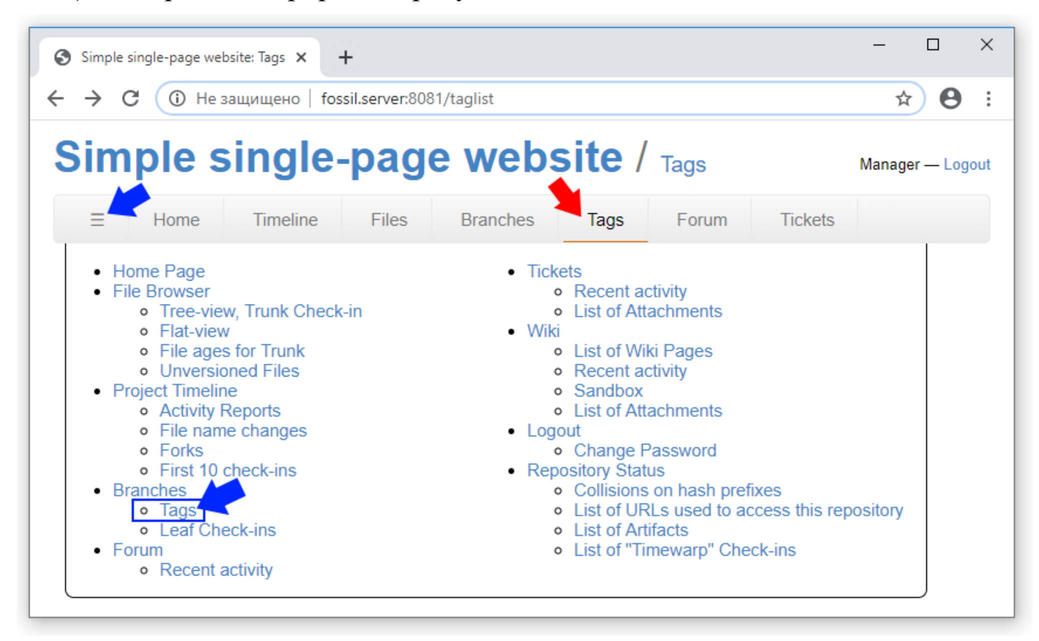


Рис. 4.30

Переход к списку нераспространяющихся ярлыков

Список ярлыков представлен гиперссылками (рис. 4.31), при переходе по которым открывается веб-страничка с теми точками сохранения на шкале времени, которым назначен соответствующий ссылке ярлык (рис. 4.32).

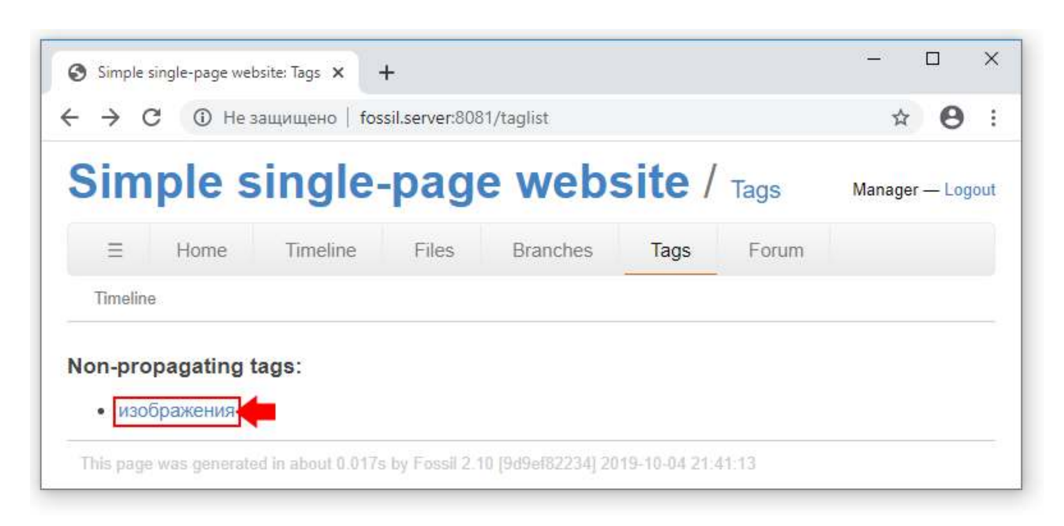


Рис. 4.31

Список нераспространяемых ярлыков

Рис. 4.32

Точки сохранения, снабжённые ярлыком изображения

А можно ли теперь использовать ярлык для обращения к точке сохранения, как это делалось раньше? И если да, то к какой из двух точек сохранения с

ярлыком изображения произойдёт такое обращение? Попробуем выполнить команду:

```
fossil ls -r "изображения"
```

Команда выполнилась успешно и вывела на экран список файлов:

```
images/demo.jpg
images/h2bkgnd.gif
images/hdrbkgnd.gif
index.css
index.html
LICENSE
```

Поскольку в списке присутствует файл h2bkgnd.gif, можно сделать вывод, что команда выполнилась над самой поздней из точек сохранения, имеющих ярлык изображения. Точно так же использование вместо идентификаторов точек сохранения названий ветвей репозитория будет приводить к воздействию команд на последние точки сохранения, имеющиеся на соответствующих ветвях.

Если ярлык был назначен ошибочно или больше не требуется, его можно убрать. В приведенном примере моментальный снимок точки сохранения 2672cdc02b содержит все три файла изображений, созданных дизайнером. Поэтому снимем ярлык изображения с ранее созданной точки сохранения 4e64309451 командой:

```
fossil tag cancel "изображения" 4e64309451
```

4.4.3. Распространяющиеся ярлыки

Мы рассмотрели использование точечных ярлыков и теперь можем переключить внимание на распространяющиеся ярлыки. Как уже было сказано выше, распространяющийся ярлык, будучи назначенным какой-либо точке сохранения, автоматически назначается и всем последующим точкам сохранения, как находящимся на этой же ветви репозитория, так и на порождённых от неё ветвях. Причём не только новым точкам, но и уже существующим на момент создания ярлыка. Рассмотрим пример.

На ветви репозитория «заголовок» верстальщик сначала оформлял текст, а в точке сохранения с идентификатором dbaf60e2bc начал использовать изображения. Назначим этой точке сохранения распространяющийся ярлык «заголовок с изображениями» с помощью команды:

```
fossil tag add --propagate "заголовок с изображениями" dbaf60e2bc
```

Параметр *--propagate* заставляет назначенный ярлык распространиться на все точки сохранения ветви «заголовок», начиная с указанной. В результате эта ветвь репозитория приобретёт вид, который можно посмотреть с помощью команды:

```
fossil tag find "заголовок"
```

Эта команда выведет следующие сведения:

```
=== 2020-03-17 ===
09:33:13 [4aec8726e4] подобран шрифт для заголовка и установлено
выравнивание
```

```
по левому краю (user: Imposer2 tags: заголовок, заголовок с
изображениями)
09:24:09 [b2d0ad5521] увеличение высоты заголовка (user: Imposer2 tags:
заголовок, заголовок с изображениями)
06:49:50 [dbaf60e2bc] подключено фоновое изображение для заголовка
(user:
Imposer2 tags: заголовок, заголовок с изображениями)
=== 2020-03-16 ===
18:55:29 [7b14ea86a2] центрирование содержимого заголовка (user: Imposer2
tags: заголовок)
18:52:17 [3d6f997e13] центрирование блока заголовка (user: Imposer2 tags:
заголовок)
=== 2020-03-13 ===
20:29:14 [dc060ff071] блок названия организации и её девиза (user: Imposer2
tags: заголовок)
17:52:44 [93b91744b7] Create new branch named "заголовок" (user:
Manager tags:
заголовок)
+++ no more data (7) +++
```

Несмотря на то, что ярлык «заголовок с изображениями» был назначен точке сохранения dbaf60e2bc, он появился и у более поздних точек сохранения b2d0ad5521 и 4aec8726e4, о чём свидетельствует информация в поле tags. Между прочим, новый ярлык не заменил ярлык «заголовок», а добавился к нему. Так что точки сохранения могут иметь несколько ярлыков.

Теперь при необходимости можно легко выбрать точки сохранения, соответствующие заголовку с изображениями:

```
fossil tag find "заголовок с изображениями"
```

Эта команда выведет на экран список из трёх точек сохранения:

```
=== 2020-03-17 ===
09:33:13 [4aec8726e4] подобран шрифт для заголовка и установлено
выравнивание
по левому краю (user: Imposer2 tags: заголовок, заголовок с
изображениями)
09:24:09 [b2d0ad5521] увеличение высоты заголовка (user: Imposer2
tags:
заголовок, заголовок с изображениями)
06:49:50 [dbaf60e2bc] подключено фоновое изображение для заголовка
(user:
Imposer2 tags: заголовок, заголовок с изображениями)
+++ no more data (3) +++
```

Распространение ярлыка, созданного с ключом *--propagate*, продолжается до тех пор, пока не будет явным образом остановлено. Предположим, что принято решение ветвь «заголовок» с точки сохранения dbaf60e2bc именовать не иначе как «заголовок с изображениями». Тогда надо в этой точке сохранения отключить распространение ярлыка «заголовок» с помощью команды:

```
fossil tag cancel "заголовок" dbaf60e2bc
```

Теперь команда fossil tag find «заголовок» выведет на экран только первые четыре точки сохранения из этой ветви:

```
=== 2020-03-16 ===
18:55:29 [7b14ea86a2] центрирование содержимого заголовка (user:
Imposer2
tags: заголовок)
18:52:17 [3d6f997e13] центрирование блока заголовка (user:
Imposer2 tags:
заголовок)
=== 2020-03-13 ===
20:29:14 [dc060ff071] блок названия организации и её девиза (user:
Imposer2
tags: заголовок)
17:52:44 [93b91744b7] Create new branch named "заголовок" (user:
Manager tags:
заголовок)
+++ no more data (4) +++
```

Но это не означает, что в репозитории появилась новая ветвь «заголовок с изображениями». Ветвей по-прежнему три, в чём можно убедиться с помощью команды fossil branch:

```
* trunk
заголовок
содержимое
```

Все действия с ярлыками, которые были выполнены, ещё до загрузки в репозиторий можно увидеть с помощью команды fossil timeline:

```
=== 2020-03-20 ===
11:29:31 [40edee66c6] Edit [dbaf60e2bcedc83e|dbaf60e2bc]: Cancel
tag "заголовок". (user: Manager)
11:26:52 [cf60fcdcec] Edit [dbaf60e2bcedc83e|dbaf60e2bc]: Add
propagating tag "заголовок с изображениями". (user:
Manager)
11:25:16 [8914cd4105] Edit [178eab2035929fec|178eab2035]: Cancel
tag "изображения". (user: Manager)
=== 2020-03-19 ===
18:23:47 [bff5765496] Edit [178eab2035929fec|178eab2035]: Add tag
"изображения". (user: Manager)
18:01:14 [c1e0e685a7] Edit [2672cdc02b453f65|2672cdc02b]: Add tag
"изображения". (user: Manager)
15:48:39 [6a77c704f6] *CURRENT* назначен ярлык "изображения" точке
сохранения с графическими элементами (user: Manager
tags: trunk)
15:15:45 [f79909b803] Edit [4e6430945143ba9e|4e64309451]: Add tag
"изображения". (user: Manager)
15:15:43 [62e089b010] Edit [4e6430945143ba9e|4e64309451]: Add tag
"изображения". (user: Manager)
=== 2020-03-18 ===
09:03:17 [9b1e92e400] *MERGE* принято содержимое веб-страницы (user:
Manager tags: trunk)
```

```
06:42:20 [a2d5807ce2] исправлен шрифт горизонтального меню в за-
головке странички (user: Manager tags: trunk)
=== 2020-03-17 ===
17:47:37 [56e4a0c8f6] *MERGE* принят заголовок веб-страницы (user:
Manager tags: trunk)
14:44:46 [fc6b890fa2] подключено фоновое изображение к заголовкам
информационных блоков (user: Imposer1 tags:
содержимое)
09:41:18 [2672cdc02b] добавлено фоновое изображение для заголовков
информационных блоков (user: Designer tags: trunk,
изображения)
--- line limit (20) reached ---
```

Такие записи имеют отметку Edit (Редактирование) и одну из расшифровок: Add tag (Добавлен ярлык), Add propagating tag (Добавлен распространяющийся ярлык) и Cancel tag (Отменён ярлык).

А что произойдёт, если отменить распространение ярлыка «заголовок» в самой первой точке сохранения одноимённой ветви репозитория? Выполним команду:

```
fossil tag cancel "заголовок" 93b91744b7
```

После этой команды никакие точки сохранения больше не отмечены ярлыком «заголовок», в чём можно убедиться с помощью команды:

```
fossil tag list
```

Она выведет на экран названия всех имеющихся в репозитории ярлыков, среди которых не будет названия «заголовок»:

```
trunk
заголовок с изображениями
изображения
содержимое
```

В то же время команда fossil branch будет, как и раньше, отображать три ветви: trunk, заголовок, содержимое.

4.4.4. Свойства

Почему же ветвь «заголовок» не пропала из списка вместе с одноимённым ярлыком или хотя бы не сменила своё название? Дело в том, что в репозитории Fossil все ярлыки, как точечные, так и распространяющиеся, являются свойствами. Свойство — это атрибут, имеющий имя и значение. В каком-то смысле свойство можно сравнить с переменной в языках программирования.

Ярлык — это свойство с именем sym-НазваниеЯрлыка и пустым значением. Чтобы посмотреть список имеющихся в репозитории свойств, можно воспользоваться командой:

```
fossil tag list --raw
```

Ключ *--raw* заставляет команду вывести на экран перечень свойств, а не ярлыков, как это делалось раньше:

```
branch
closed
```

```
sym-trunk
sym-заголовок
sym-заголовок с изображениями
sym-изображения
sym-содержимое
```

Чтобы переименовать ветвь репозитория, надо свойству branch первой точки сохранения на этой ветви присвоить значение нового имени. Например, чтобы все ветви были обозначены русскими словами, можно ветвь trunk переименовать в «основная». Для этого достаточно выполнить команду:

```
fossil tag add --raw --propagate branch 24e23c5cfb "основная"
```

Важно указать ключ *--raw*, потому что без него вместо присвоения значения «основная» свойству branch будет создан обыкновенный ярлык branch (свойство sym-branch без значения).

После переименования команда fossil branch выводит названия ветвей так:

```
заголовок
* основная
содержимое
```

Однако все точки сохранения, находящиеся на ветви «основная», попрежнему снабжены ярлыками trunk. И заменить их на ярлыки «основная» не так просто. Отмена ярлыков trunk реализуется командой:

```
fossil tag cancel "trunk" 24e23c5cfb
```

Но если для назначения новых ярлыков «основная» выполнить приведенную ниже команду, то эти ярлыки получат также точки сохранения на ветвях «заголовок» и «содержимое», выращенных позже на основной ветви trunk в ходе развития проекта.

```
fossil tag add --propagate "основная" 24e23c5cfb
```

Чтобы восстановить логику, придётся выполнить команды прерывания распространения ярлыка «основная» в первых точках сохранения, находящихся на ответвлениях:

```
fossil tag cancel "основная" 93b91744b7
fossil tag cancel "основная" ddc3084089
```

Результат переименования можно записать в репозиторий командой:

```
fossil commit --allow-empty -m "переименована ветвь \"trunk\" в
\"основная\""
```

После переименования основная ветвь изменит свой цвет на шкале времени. Но эта проблема поправима. Устанавливать фоновый цвет для точек сохранения на шкале времени можно с помощью свойства bgcolor. Например, чтобы последняя точка сохранения с ярлыком «изображения» отображалась на пурпурном фоне, надо ввести команду:

```
fossil tag add --raw bgcolor "изображения" magenta
```

А если попытаться перекрасить точки сохранения ветви «основная» с помощью приведенной ниже команды, то получится неприятный сюрприз: лазурный фон распространится и на ответвления.

```
fossil tag add --raw --propagate bgcolor 24e23c5cfb cyan
```

Исправить ситуацию можно установкой в первых точках сохранения ответвлений явного запрета на распространение этого свойства:

```
fossil tag cancel --raw bgcolor 93b91744b7
fossil tag cancel --raw bgcolor ddc3084089
```

Результаты переименований и перекрашиваний надо записывать в репозиторий командой fossil commit с ключом *--allow-empty*. После проделанных операций шкала времени в веб-интерфейсе приобретёт вид, показанный на рисунке 4.33.

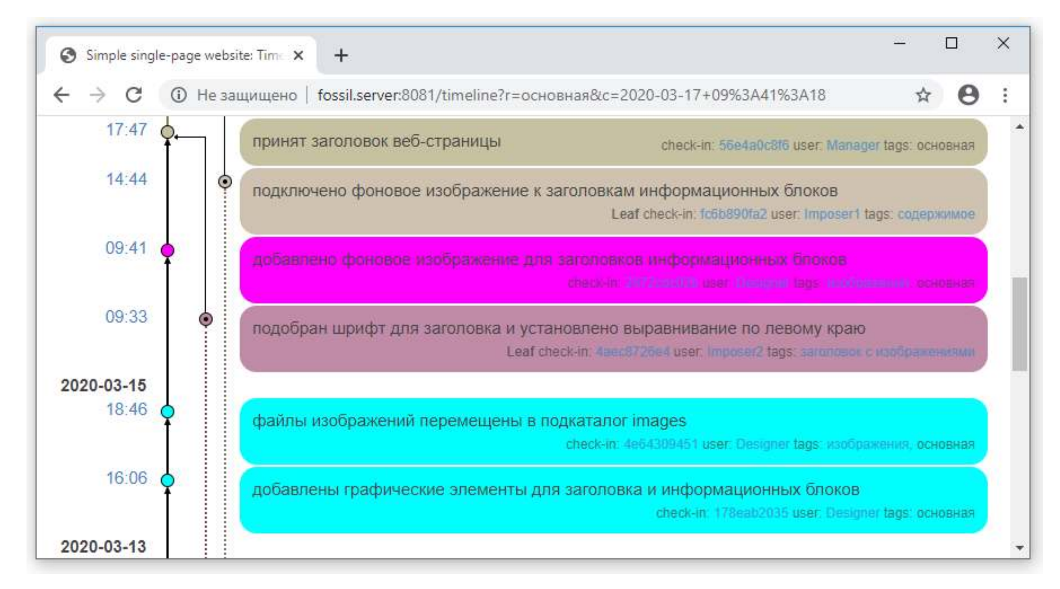


Рис. 4.33

Шкала времени после переименований и перекрашиваний

Несмотря на то, что путём модификации значений свойств branch и bgcolor можно переименовать и перекрасить ветви репозитория, во избежание лишней работы и побочных эффектов лучше давать им сразу предпочтительные названия и цвет при ответвлении с помощью команды:

```
fossil branch new --bgcolor ЦветФона НазваниеВетви ТочкаОтветвления
```

В приведённой команде ключ *--bgcolor* позволяет указать фоновый цвет для точек сохранения ветви, который будет использован вместо цвета, автоматически предоставляемого системой Fossil. Завершают команду два параметра — собственно название ветви и идентификатор (или ярлык) точки сохранения, в которой создаётся ответвление.

Система Fossil позволяет использовать одинаковые имена для разных ветвей. В этом случае при указании названия ветви в командах они будут применяться к той ветви, на которой находится самая последняя точка сохранения.

4.4.5. Исправление значений свойств

Возможность изменения свойств элементов репозитория придаёт системе гибкость, однако пользоваться этой возможностью следует с осторожностью и по необходимости. Например, после переименования основной ветви репозитория из «trunk» в «основная» команда открытия репозитория на новом рабочем месте fossil open ..\sspwebsite-local.fossil перестанет работать и начнёт сообщать об ошибке:

```
not found: trunk
```

Дело в том, что эта команда по умолчанию ищет последнюю точку сохранения на ветви с именем trunk, а теперь такая ветвь в репозитории отсутствует. Для того, чтобы открыть репозиторий в новых условиях, придётся явным образом указывать в команде название ветви:

```
fossil open ..\sspwebsite-local.fossil "основная"
```

Таким образом, перед внесением изменений стоит подумать, перевешивает ли прогнозируемый результат те проблемы, с которыми придётся столкнуться. С другой стороны, не исключены ситуации, когда при создании новой ветви в её названии была просто допущена опечатка, которую заметили лишь спустя некоторое время. В этом случае, конечно, ошибку лучше исправить.

Но для этого есть более простой способ, чем тот, который описан выше. Например, вернуть основной ветви репозитория первоначальное название позволяет команда:

```
fossil amend 24e23c5cfb --branch trunk
```

В этой команде 24e23c5cfb — это идентификатор точки сохранения, с которой начинается ветвь, требующая переименования, а с помощью параметра *--branch* указывается требуемое имя. В случае успеха команда выведет следующее сообщение:

```
uuid: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29 07:24:50 UTC
tags: trunk
comment: initial empty check-in (user: SetupUser)
```

В первой строке указано полное значение идентификатора точки сохранения и время её создания, во второй строке — ассоциированные с ней ярлыки, а в третьей — её описание. Остаётся лишь загрузить выполненную операцию в репозиторий командой:

```
fossil commit --allow-empty -m "Основной ветви возвращено первона-
чальное имя"
```

Очевидно, что использование команды fossil amend гораздо удобнее, чем серии манипуляций с изменением значений свойств. Поэтому рассмотрим ещё один полезный пример её применения. Ветви создаются не так часто по сравне-

нию с записью в репозиторий мгновенных снимков рабочих каталогов. Команды fossil commit набираются разработчиками практически в автоматическом режиме, и тем досаднее ошибки, которые легко пропустить в описании выполненных изменений. К счастью, их можно легко исправить, воспользовавшись только что рассмотренным приёмом.

Если теперь уже стало понятно, что переименовывать основную ветвь — не самая лучшая идея, то изменение описания первой точки сохранения не должно отразиться на работоспособности репозитория. Поэтому выполним команду:

```
fossil amend 24e23c5cfb -m "начальная пустая точка сохранения" --
author Manager
```

Как и в случае переименования ветви, сначала указывается идентификатор точки сохранения, на которую действует команда — 24e23c5cfb. А затем с помощью параметра *-m* указывается текст нового описания. Заодно можно изменить имя пользователя, создавшего точку сохранения. Для этого служит параметр *--author*. В результате выполнения команда вывела обновлённые сведения о точке сохранения.

```
uuid: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29 07:24:50 UTC
tags: trunk
comment: начальная пустая точка сохранения (user: Manager)
```

Чтобы внесённые изменения сразу отразились в репозитории, выполним их загрузку:

```
fossil commit --allow-empty -m "исправлен комментарий начальной
точки сохранения"
```

В результате произведенных манипуляций начальная точка сохранения в репозитории приобрела вид, показанный на рисунке 4.34.

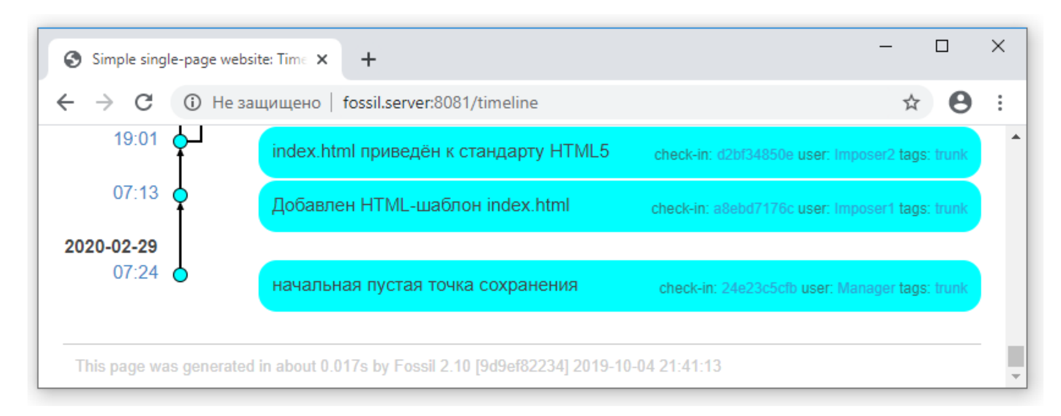


Рис. 4.34

Начальная точка сохранения с новым автором, описанием и восстановленным названием ветви

Оказывается, выполнить все описанные действия можно через веб-интерфейс. Для этого потребуется отыскать на шкале времени точку сохранения, в свойства которой надо внести изменения (что в описываемом примере не составляет труда), перейти по гиперссылке с её идентификатором, обозначенным check-in, и в открывшемся окне воспользоваться гиперссылкой edit в строке Other Links: (рис. 4.35).

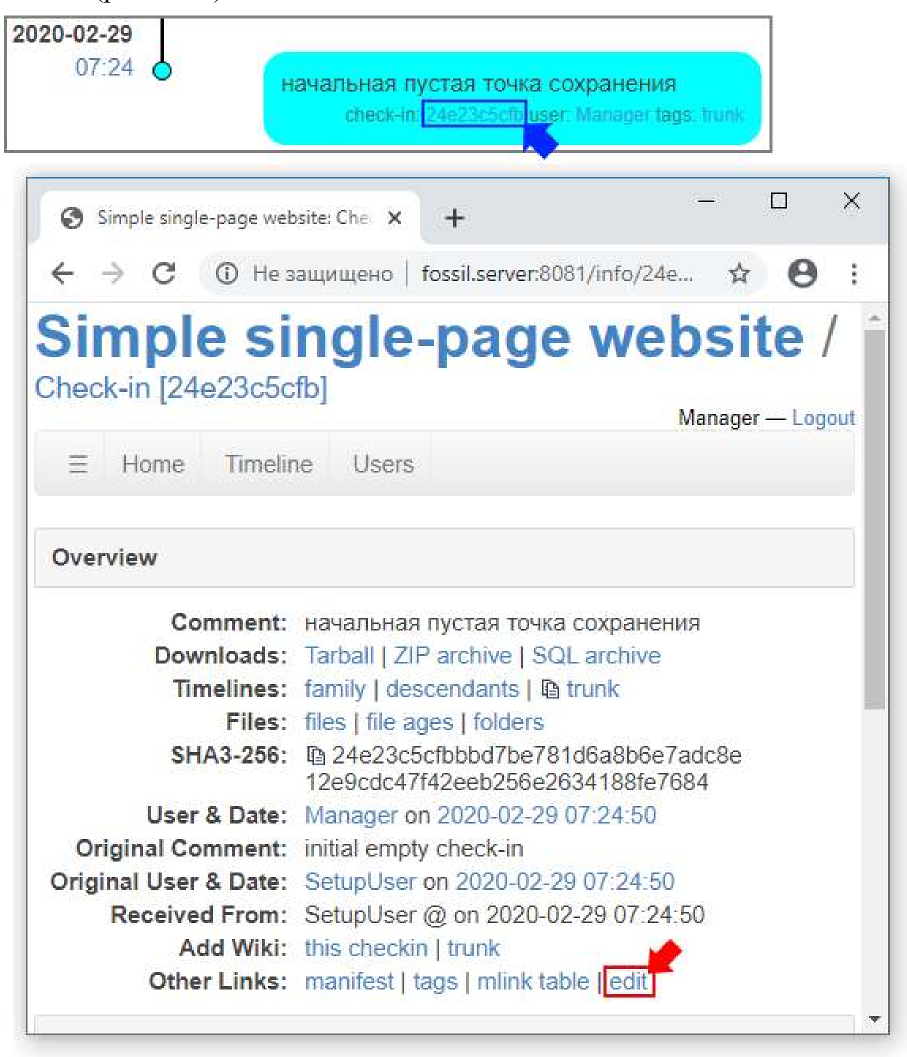


Рис. 4.35

Переход к форме редактирования свойств точки сохранения

В окне веб-браузера откроется форма изменения свойств выбранной точки сохранения (рис. 4.36), поля которой заполнены текущими, недавно установленными, значениями. Рассмотрим элементы управления этой формы более подробно.

Рис. 4.36

Форма изменения свойств точки сохранения

В строке редактирования User можно указать идентификатор пользователя репозитория, который будет назначен автором этой точки сохранения. В текстовое поле Comment можно ввести новое описание точки сохранения. В строке редактирования Check-in Time указаны дата и время создания точки сохранения. При необходимости это значение тоже можно изменить.

Группа элементов управления Background Color позволяет изменить цвет фона выбранной точки сохранения. Для этого надо установить флажок Change background color и выбрать желаемый цвет из палитры, которая появляется при щелчке по образцу цвета, отображаемому справа от флажка. Установка флажка Propagate color to descendants приведёт к распространению выбранного цвета на всех потомков этой точки сохранения.

Группа элементов управления Tags: предназначена для изменения ярлыков точки сохранения. Если установить флажок Add the following new tag name to this check-in, то точке сохранения будут назначены ярлыки, перечисленные в строке ввода справа от этого флажка. С помощью флажков Cancel special tag bgcolor и

Cancel special tag user можно прекратить действие специальных ярлыков bgcolor (цвет фона) и user (имя пользователя) соответственно.

Чтобы изменить название ветви, которую начинает редактируемая точка сохранения, надо установить флажок Branching: и ввести новое название в строку редактирования, расположенную справа от него.

Установка флажка Branch Hiding приведёт к тому, что эта и последующие точки сохранения на текущей ветви не будут отображаться на шкале времени. Это может быть полезно для сокрытия ветвей, которые были созданы с целью проверки идей, впоследствии признанных бесперспективными.

И, наконец, установка флажка Branch Closure приведёт к закрытию текущей ветви, т. е. запрету дальнейшей записи в неё мгновенных снимков рабочего каталога.

После заполнения формы надо нажать кнопку Preview (Предпросмотр), в результате чего в верхней части формы на фоне выбранного цвета отобразится новый текст комментария с новым именем пользователя и новыми тегами, а в нижней части формы появится кнопка Apply Changes (Применить изменения), с помощью которой выполненные изменения могут быть применены к точке сохранения.

Глава 5
СИСТЕМА УЧЁТА ЗАЯВОК НА ДОРАБОТКУ

5.1. Требования к разрабатываемому продукту

Рассмотренная в предыдущих главах система контроля версий является одним из важнейших, но не единственным инструментом, улучшающим процесс разработки. Она упорядочивает только один его аспект — работу с исходными текстами разрабатываемого продукта. Но есть и другие, например работа с требованиями к системе.

При проектировании системы составляется список требований, которым она должна удовлетворять. Требования могут быть функциональными (перечень функций, которые система должна реализовывать) и нефункциональными (минимальные уровни производительности, точности, надёжности и т. п.). В ходе разработки продукт проверяется на соответствие предъявляемым к нему требованиям. Когда все предъявленные требования будут выполнены, программа считается готовой к выпуску в опытно-промышленную эксплуатацию.

Редко бывает, чтобы выпущенная в эксплуатацию система не имела никаких недостатков. Возможно, какое-то из требований было упущено на этапе контроля качества. Может проявиться какой-нибудь просчёт, не учтённый при проектировании. Случается, что какие-то нефункциональные требования надо ужесточить, чтобы улучшить эксплуатационные характеристики системы.

Недостатки после их обнаружения должны быть сформулированы, оформлены как претензии или пожелания и переданы разработчику программного продукта в виде заявок на доработку. В системе Fossil имеется подсистема для учёта таких заявок. Несмотря на то, что взаимодействовать с этой подсистемой можно из командной строки с помощью команд fossil ticket..., удобнее работать с ней через веб-интерфейс.

Чтобы приступить к работе с заявками на доработку, надо выбрать пункт Tickets в горизонтальном или раскрывающемся меню (рис. 5.1) или ввести в адресной строке веб-браузера URL: http://fossil.server:8081/ticket, где fossil.server — сетевое имя сервера, а 8081 — номер TCP-порта, на котором система Fossil принимает сетевые подключения.

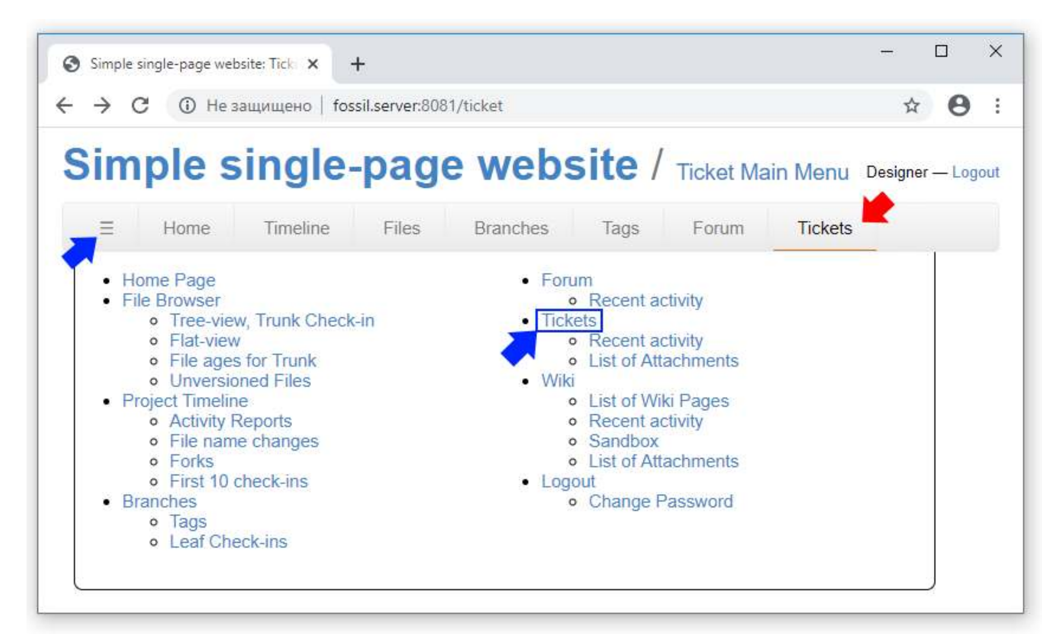


Рис. 5.1

Переход к работе с заявками на доработку

5.2. Создание заявки на доработку

Продолжим рассмотрение примера разработки одностраничного веб-сайта небольшой командой специалистов. Основная часть работы к этому времени уже выполнена, и менеджер попросил дизайнера проверить веб-страничку на соответствие макету. Первым делом дизайнер выгрузил из репозитория в свой рабочий каталог последний вариант файлов проекта с помощью команды:

```
fossil update
```

Затем он открыл файл index.html в веб-браузере и начал проверку. На первый взгляд всё выглядело неплохо. Но стоило ему уменьшить размер окна, как иллюстрации информационных блоков наехали на текст и сплющили его в узенькие колонки.

Дизайнер понимал, что верстальщикам это было трудно предвидеть по статическому макету, но такая реализация веб-странички была неприемлемой — она бы отпугнула от неё пользователей мобильных устройств. Поэтому он открыл веб-интерфейс на страничке заявок на доработку и выбрал гиперссылку New ticket (Новая заявка) (рис. 5.2).

В веб-браузере открылась форма для заполнения, показанная на рисунке 5.3.

Форма состоит из семи полей. В первом поле Enter a one-line summary of the ticket: предлагается ввести краткое описание заявки одной строкой.



Рис. 5.2

Создание новой заявки на доработку

Simple single-page website: New

Не защищено | fossil.server:8081/tktnew

Simple single-page website / New Ticket

Designer — Logout

Home | Timeline | Files | Branches

Reports

Enter A New Ticket

Enter a one-line summary of the ticket:
Не масштабируются иллюстрации информационных блоков

Type: Feature_Request — What type of ticket is this?

Version: b22fa79596 — In what version or build number do you observe the problem?

Severity: Cosmetic — How debilitating is the problem? How badly does the problem affect the operation of the product?

EMail: designer@fossil.server — Not publicly visible Used by developers to contact you with questions.

Enter a detailed description of the problem. For code defects, be sure to provide details on exactly how the problem can be reproduced. Provide as much detail as possible. Format: Plain Text

Иллюстрации информационных блоков не уменьшаются при уменьшении размеров окна веб-браузера, что приводит к искажению отображения текста. Надо, чтобы размеры иллюстраций всегда были пропорциональны видимым размерам информационных блоков.

Preview — See how the description will appear after formatting.

Cancel — Abandon and forget this ticket

This page was generated in about 0.017s by Fossil 2.10 [9d9ef82234] 2019-10-04 21:41:13

Рис. 5.3

Форма для заполнения новой заявки

Во втором поле Type: предлагается выбрать тип заявки из следующих вариантов:

- Code_Defect — ошибка в коде;
- Build_Problem — проблема сборки;
- Documentation — документация;
- Feature_Request — запрос на доработку;
- Incident — инцидент, проблема с безопасностью.

В третьем поле Version: надо ввести информацию о версии продукта, к которой относится создаваемая заявка.

В четвёртом поле Severity: предлагается указать степень серьёзности проблемы, которая послужила причиной создания заявки. На выбор даются следующие варианты:

- Critical — критическая, продукт невозможно или недопустимо использовать до устранения проблемы;
- Severe — серьёзная, использование продукта с этой проблемой может привести к неприятностям;
- Important — важная, проблема должна быть устранена в самое ближайшее время;
- Minor — незначительная, заявку надо выполнить по мере возможности;
- Cosmetic — косметическая, пожелание по поводу улучшения системы.

В пятом поле EMail: надо ввести адрес электронной почты автора заявки. Несмотря на то, что доступ к подсистеме управления заявками есть у многих пользователей, введённая в это поле информация будет доступна только разработчикам. С её помощью они смогут уточнить вопросы, которые могут появиться при работе над заявкой.

В последнем большом текстовом поле предлагается ввести подробное описание проблемы, которая явилась причиной создания заявки. Здесь желательно написать о том, как описанную проблему можно воспроизвести (ведь чтобы её устранить, надо видеть, что устранять).

Над текстовым полем находится поле выбора из списка способа, каким будет оформлен текст описания проблемы. Доступны следующие варианты:

- Wiki — разметка, принятая для написания статей в Wiki;
- HTML — гипертекстовая разметка для веб-страниц;
- Plain Text — обычный текст;
- [links only] — текст с распознаванием ссылок на сущности системы Fossil в виде [Идентификатор].

Дизайнер заполнил поля формы по возникшему у него вопросу и нажал кнопку Preview (Предпросмотр). В окне отобразился текст набранного им описания проблемы и появилась кнопка Submit (Отправка). Проверив текст описания и не обнаружив в нём ошибок, дизайнер нажал кнопку Submit. В окне отобразилась карточка заявки, созданной дизайнером (рис. 5.4).

Рис. 5.4

Карточка созданной заявки

5.3. Прикрепление поясняющих материалов

Первым эту заявку увидел менеджер во время очередного обхода системы учёта заявок. Он зашёл на веб-страницу этой системы и перешёл по гиперссылке All Tickets (рис. 5.5).

В веб-браузере отобразилась таблица с заявками на доработку, в которой помимо заголовка присутствует единственная строка, соответствующая созданной дизайнером заявке (рис. 5.6). Таблица состоит из семи колонок. В первой колонке находится гиперссылка с идентификатором заявки, во второй — дата и время её создания (изменения), в третьей — тип заявки, в четвёртой — её состояние, в пятой — подсистема, к которой она относится, в шестой — краткое описание. И, наконец, в седьмой колонке размещена гиперссылка edit, по которой можно перейти в режим редактирования заявки.

Simple single-page website: Ticke ×

Не защищено | fossil.server:8081/reportlist

Simple single-page website / Ticket Main Menu

Manager — Logout

☰ Home Timeline Files Branches Users

New

Enter a new ticket:

- New ticket

Choose a report format from the following list:

1. All Tickets [copy] [edit][sql]

Other options:

- New report format
- Tend to pending moderation requests

This page was generated in about 0.034s by Fossil 2.10 [9d9ef82234] 2019-10-04 21:41:13

Рис. 5.5

Переход на список заявок

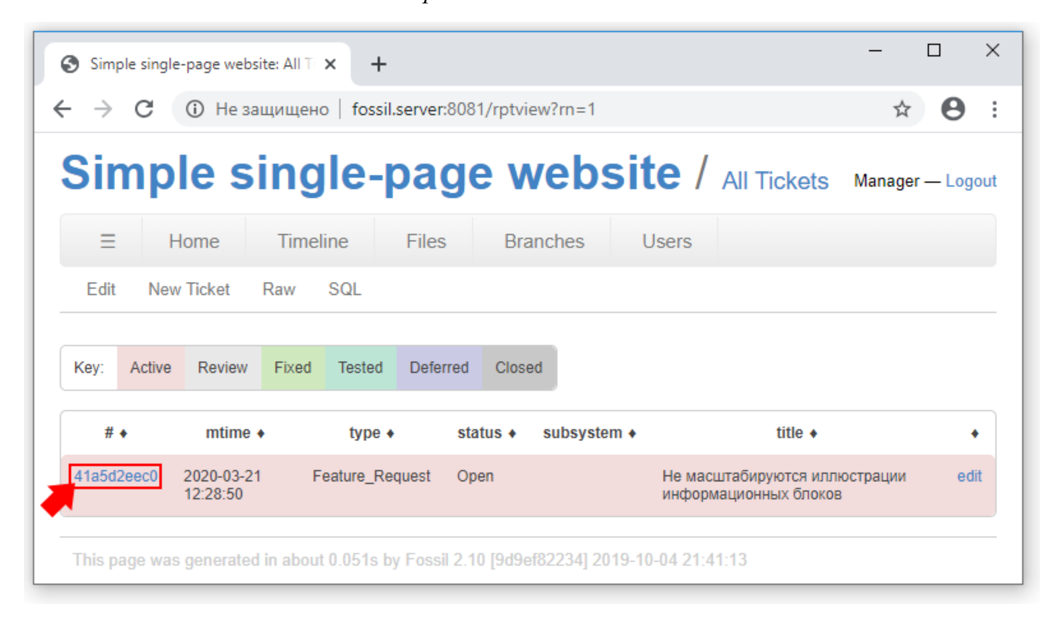


Рис. 5.6

Таблица с заявками

Чтобы ознакомиться с подробным описанием заявки, менеджер перешёл по гиперссылке 41a5d2eec0, расположенной в первой колонке. Открылась карточка заявки, содержащая всю информацию, введённую при её создании. Менеджер открыл в веб-браузере файл index.html из рабочего каталога и убедился, что описанный в заявке недостаток действительно присутствует.

Для наглядности менеджер решил добавить к заявке иллюстрацию, на которой демонстрируется изложенная в ней проблема. Он подготовил файл со снимком экрана, а затем выбрал пункт Attach (Прикрепить) в дополнительном горизонтальном меню карточки заявки, которое расположено между содержимым карточки и главным меню системы Fossil (рис. 5.7).

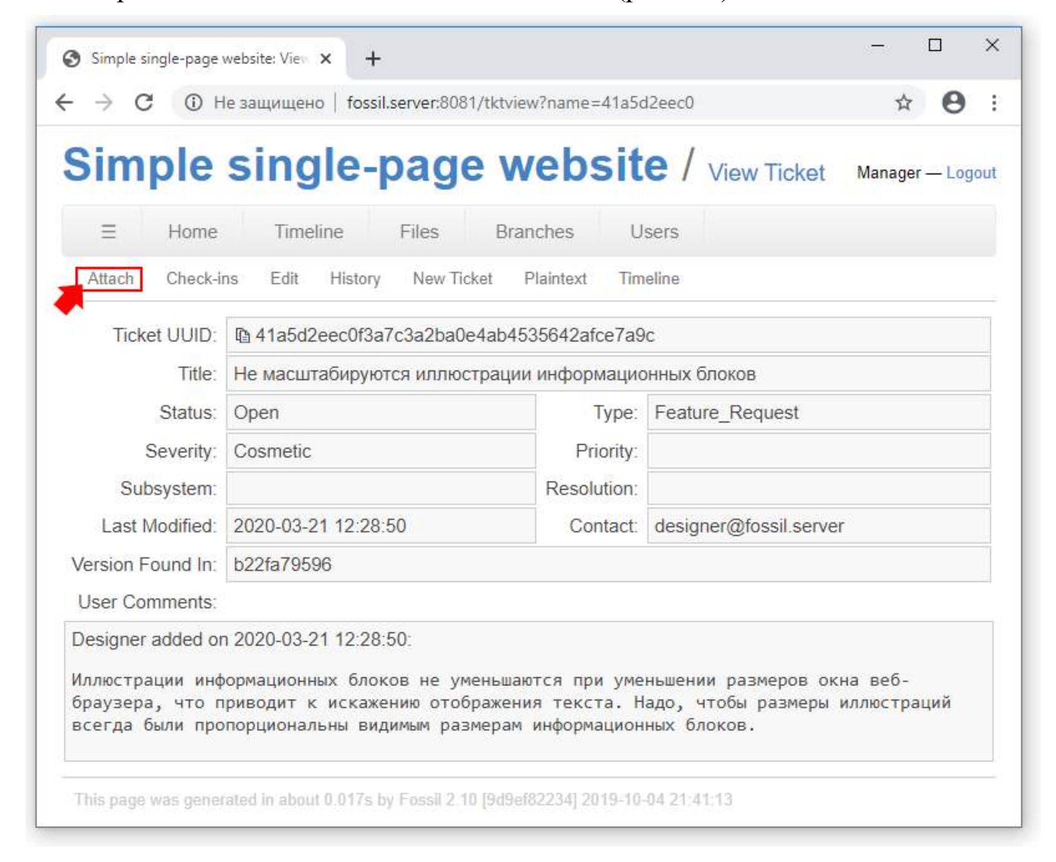


Рис. 5.7

Переход к форме присоединения файла к заявке

В открывшейся форме менеджер нажал кнопку Выберите файл, в появившемся стандартном окне диалога указал на файл со снимком экрана и нажал кнопку Открыть (рис. 5.8). Затем в поле Description: (Описание) набрал текст, поясняющий назначение прикрепляемого файла. Когда всё было сделано, он нажал кнопку Add Attachment (Добавить Вложение).

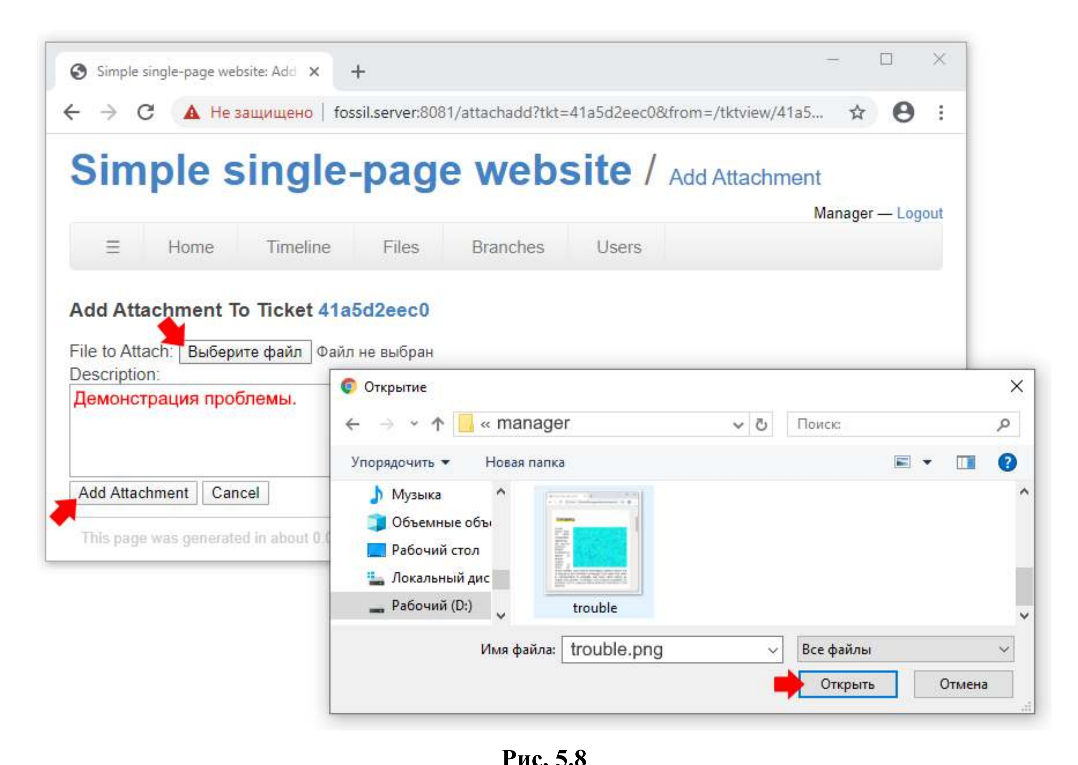


Рис. 5.8

Присоединение файла к заявке

После выполненных действий в нижней части карточки заявки появился раздел Attachments: (Вложения), в котором приведены сведения о прикреплённом файле и имеется гиперссылка [details] (подробности), позволяющая посмотреть изображение снимка экрана (рис. 5.9).

Но почему добавлением снимка экрана занимается менеджер? Почему это не сделал дизайнер самостоятельно? Дело в том, что у дизайнера не было полномочий для выполнения этой операции, у него даже отсутствовал пункт меню Attach в горизонтальном меню карточки заявки.

Чтобы пользователь Fossil мог прикреплять файлы к заявкам на доработку, в карточке его учётной записи должна стоять отметка Attachments (Вложения). Она добавляет букву b к строке разрешённых действий пользователя учётной записи (рис. 5.10).

На шкале времени в веб-интерфейсе Fossil менеджер отыскал точку сохранения, в которой было задействовано изображение для иллюстрации информационных блоков, и навёл курсор мыши на узел, соответствующий этой точке сохранения. Всплыло окно, в котором отобразился идентификатор точки сохранения. Менеджер щёлкнул левой кнопкой мыши по пиктограмме, изображённой слева от идентификатора, и идентификатор сохранился в буфере обмена (рис. 5.11).

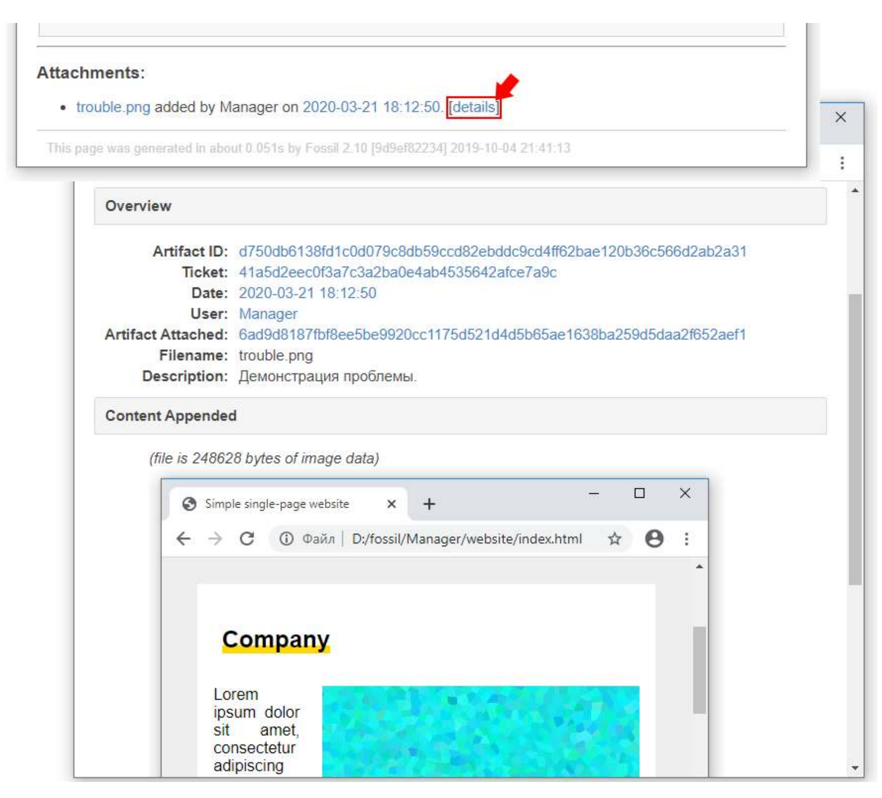


Рис. 5.9

Просмотр прикреплённого к заявке файла

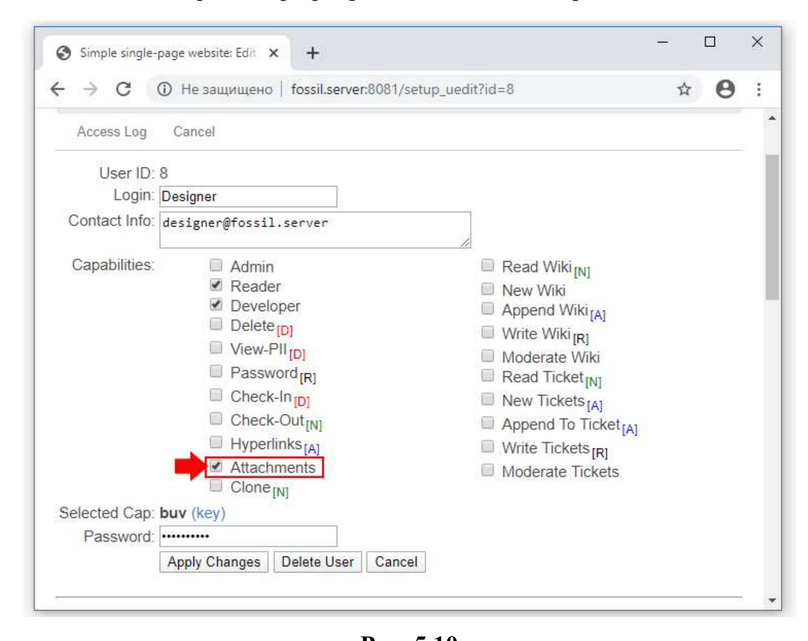


Рис. 5.10

Предоставление разрешения на прикрепление файлов

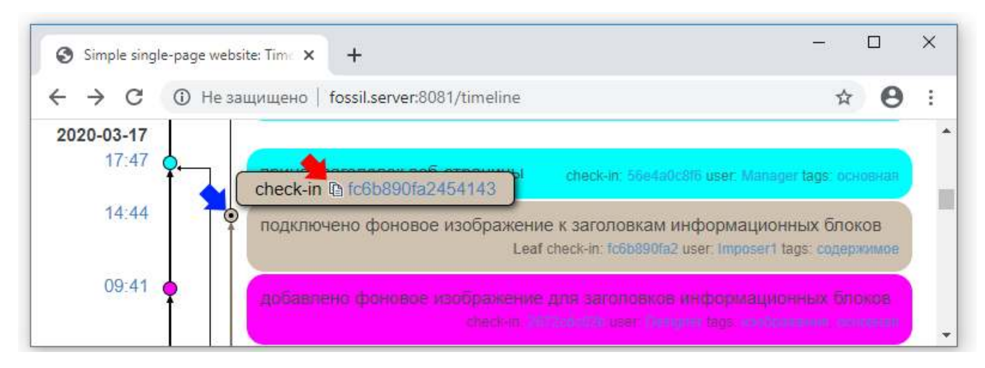


Рис. 5.11

Копирование идентификатора точки сохранения на шкале времени в буфер обмена

Затем менеджер вернулся к карточке заявки и выбрал пункт меню Edit. Открылась форма редактирования заявки, в которой многие поля повторяют те, что были на форме заполнения заявки, но есть несколько новых.

Поле Status: (состояние) позволяет выбрать вариант, отражающий этап существования заявки, из списка:

- Open — только что создана;
- Verified — проверена;
- Review — на рассмотрении;
- Deferred — исполнение отложено;
- Fixed — доработка выполнена;
- Tested — доработка проверена;
- Closed — закрыта.

В поле Priority: (Приоритет) можно указать срочность выполнения работ по заявке, выбрав из списка одно из значений:

- Immediate — выполнить прямо сейчас;
- High — сделать как можно быстрее;
- Medium — сделать в рабочем режиме;
- Low — сделать по мере возможности;
- Zero — выполнять не обязательно.

Поле Resolution: предназначено для указания решения, принятого в отношении описанной в заявке ситуации. В нём может быть выбрано одно из следующих значений:

- Open — заявка находится в работе;
- Fixed — заявка выполнена, проблема устранена;
- Rejected — заявка отклонена;
- Workaround — предложено обходное решение, костыль;
- Unable_To_Reproduce — описанную в заявке проблему не удаётся воспроизвести;
- Works_As_Designed — система работает именно так, как было задумано;
- External_Bug — причина проблемы находится за пределами системы;
- Not_A_Bug — то, что описано в заявке, не является проблемой;
- Duplicate — заявка дублирует ранее поданную;

- Overcome_By_Events — отношение к описанной в заявке проблеме изменилось в результате произошедших событий;
- Misconfiguration — проблема в настройках системы.

Менеджер поставил отметку Verified в поле Status, значение Medium в поле Priority и написал поручение верстальщику Imposer1 разобраться с описанной в заявке проблемой. К тексту поручения он добавил сохранённый в буфере обмена идентификатор точки сохранения, заключив его в квадратные скобки. Чтобы такая ссылка на точку сохранения сработала, он проверил установку формата добавляемого сообщения в значение [links only], потому что при использовании формата [Plain Text] введённое значение не было бы выделено из остального текста (рис. 5.12).

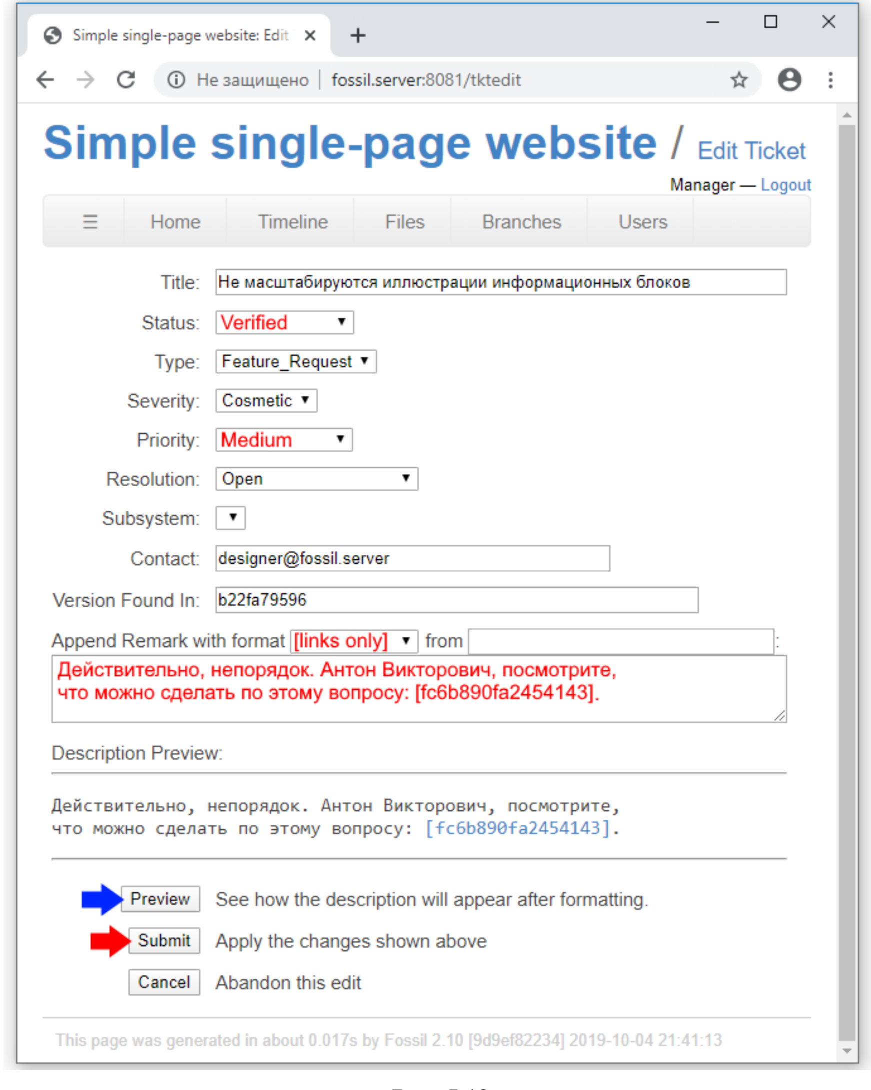


Рис. 5.12

Форма редактирования заявки

Завершил редактирование менеджер нажатием кнопки Preview (Предпросмотр) и появившейся после этого кнопки Submit (Подтверждение). В результате состояние заявки изменилось на Verified, а на карточке заявки в секции User Comments: (Комментарии Пользователей) появился текст поручения менеджера с активной гиперссылкой [fc6b890fa2454143], по которой можно сразу попасть на соответствующую точку сохранения. Более того, у точки сохранения на шкале времени появилась отметка о том, что в системе учёта заявок на доработку имеется соответствующая ей задача (рис. 5.13).

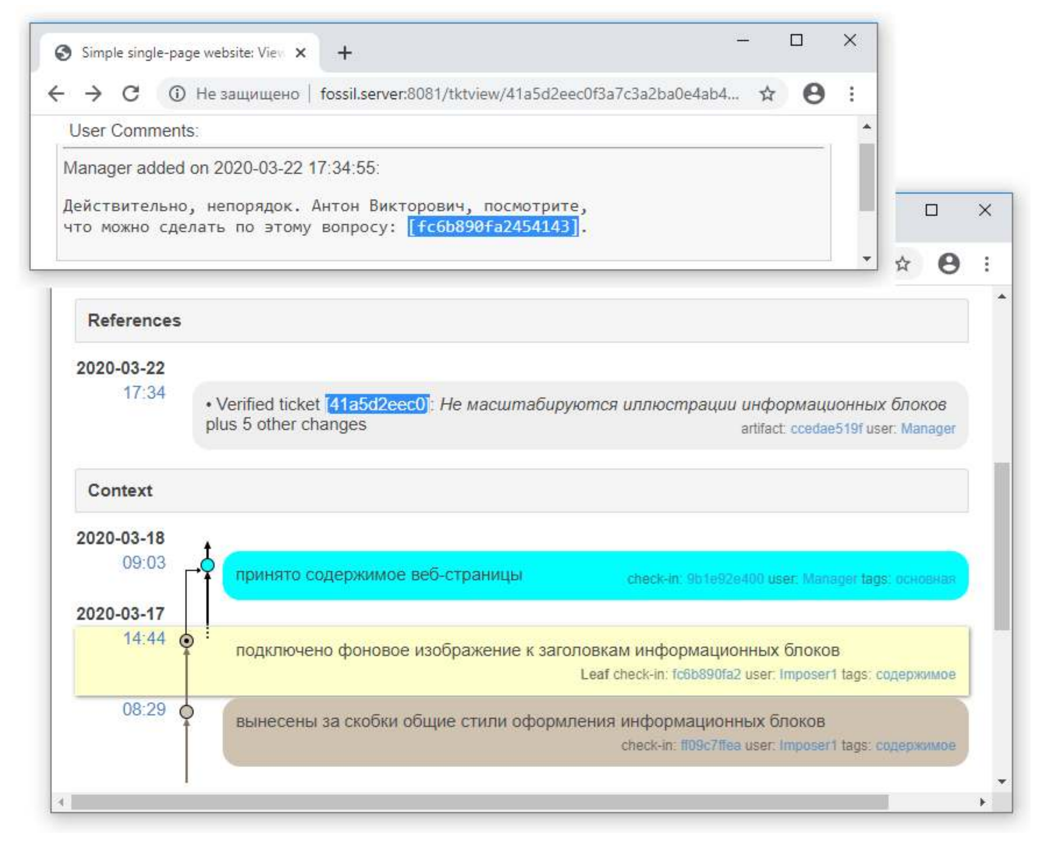


Рис. 5.13

Образовавшаяся связь между заявкой на доработку и точкой сохранения

5.4. Выполнение заявки

Первый верстальщик, которому менеджер поручил разобраться с проблемой, обновил свой рабочий каталог, убедился в том, что изображения действительно не масштабируются, и принял заявку в работу. Последнее действие он отразил в системе учёта заявок следующим образом. Он открыл карточку заявки, выбрал в меню пункт Edit и в открывшейся форме внёс два изменения:

1) в поле Status: установил значение Review;

2) в текстовом поле набрал сообщение: «Принял заявку в работу».

Затем сохранил внесённые изменения с помощью кнопок Preview и Submit.

После этого верстальщик приступил к выполнению заявки. Для того чтобы иллюстрации масштабировались вместе с изменением размера окна веб-браузера, он в файле index.css добавил правило width: 46% для селектора section img. Убедившись, что иллюстрации стали вести себя более рационально, он загрузил выполненные изменения в репозиторий проекта с помощью команды:

```
fossil commit -m "задан масштаб для иллюстраций информационных блоков"
```

В репозитории появилась точка сохранения с идентификатором be46febc61. Чтобы отчитаться о проделанной работе, верстальщик открыл в веб-браузере карточку заявки на доработку, перешёл в режим её редактирования и внёс следующие изменения:

1) в поле Status: установил значение Fixed;

2) в текстовом поле набрал сообщение: «Проблема устранена: [be46febc61]».

После того, как верстальщик сохранил внесённые в карточку изменения, система Fossil автоматически добавила к точке сохранения отметку о том, что она появилась в результате решения проблемы, описанной в заявке с идентификатором 41a5d2eec0 (рис. 5.14). Это произошло благодаря тому, что верстальщик в текстовом сообщении, дописанном к заявке, указал в квадратных скобках идентификатор точки сохранения.

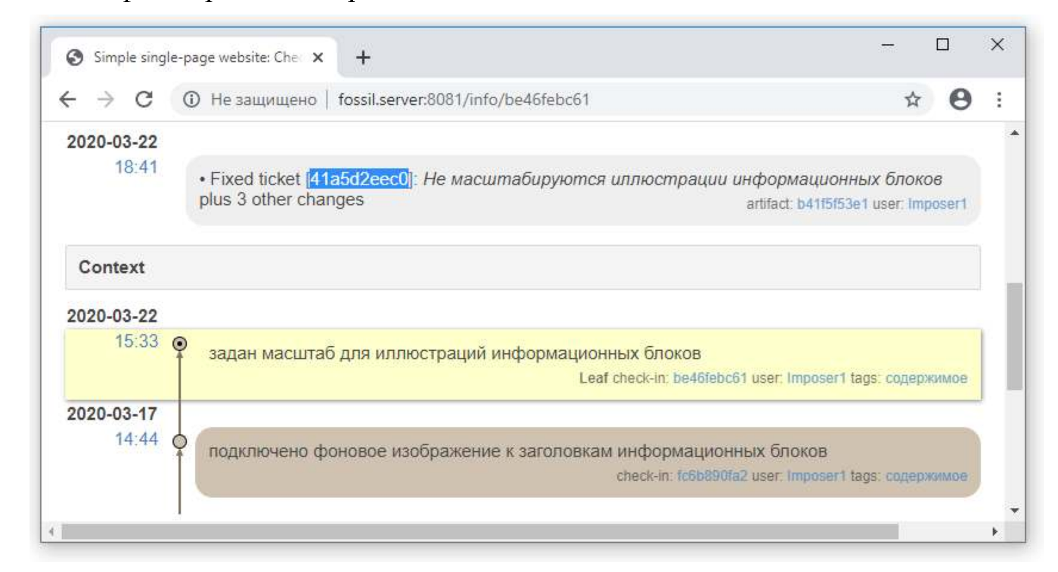


Рис. 5.14

Привязка точки сохранения на шкале времени к ассоциированной с ней заявке на доработку

Узнав о выполненной доработке, дизайнер обновил свой рабочий каталог (в результате этого получил изменённый файл index.css), убедился в работоспособности веб-страницы и незамедлительно отразил свои впечатления в системе

учёта заявок. Для этого он открыл заявку для редактирования и внёс в неё следующие изменения:

1) в поле Status: установил значение Tested;

2) в текстовом поле набрал сообщение: «Проверено. Всё работает».

Заключительный шаг работы над заявкой выполнил менеджер. Он увидел, что автор заявки — дизайнер — удовлетворён тем, как её выполнил первый верстальщик, и изменил значение в поле Status: на Closed, т. е. закрыл заявку.

А ещё, поскольку верстальщик записал проделанную работу в свою ветку репозитория (в рамках продолжения работы над содержимым веб-страницы), менеджер произвёл слияние этой ветки с основной ветвью проекта. И выполнение этой задачи ему упростило то, что верстальщик указал идентификатор точки сохранения, куда записано решение проблемы, в комментарии к заявке на доработку:

```
fossil merge be46febc61
```

При выполнении команды никаких проблем не возникло:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 476 received: 453 ip: 192.168.56.101
MERGE index.css
"fossil undo" is available to undo changes to the working checkout.
```

Перед записью в репозиторий мгновенного снимка каталога, образовавшегося в результате слияния ветвей, менеджер выяснил идентификатор заявки на доработку с помощью команды:

```
fossil ticket show 1 status=='Closed'
```

В приведенной команде 1 — это номер отчёта о заявках. Перечень возможных отчётов можно получить с помощью команды fossil ticket list reports.

Завершается команда условием, которому должны удовлетворять отбираемые в отчёт записи. В данном случае значение поля status (состояние) должно быть равно Closed (Закрыт), т. е. будут отобраны только закрытые заявки на доработку. Список полей, с которыми можно составлять условия, выводит команда fossil ticket list fields.

В результате выполнения команды на экран была выведена следующая информация:

```
bgcolor # mtime type status subsystem title
#c8c8c8 41a5d2eec0 2020-03-22 19:28:06 Feature_Request Closed Не
масштабируются иллюстрации информационных блоков
```

Здесь есть все сведения, которые присутствуют в таблице заявок, доступной через веб-интерфейс, правда, в менее презентабельном виде, который ограничен изобразительными возможностями интерфейса командной строки.

Идентификатор закрытой заявки в квадратных скобках менеджер указал в описании точки сохранения, которую создал в репозитории командой:

```
fossil commit -m "принято масштабирование иллюстраций по заявке
[41a5d2eec0]"
```

На шкале времени появилась точка сохранения, в описании которой имеется гиперссылка, позволяющая быстро переключиться на просмотр заявки (рис. 5.15).

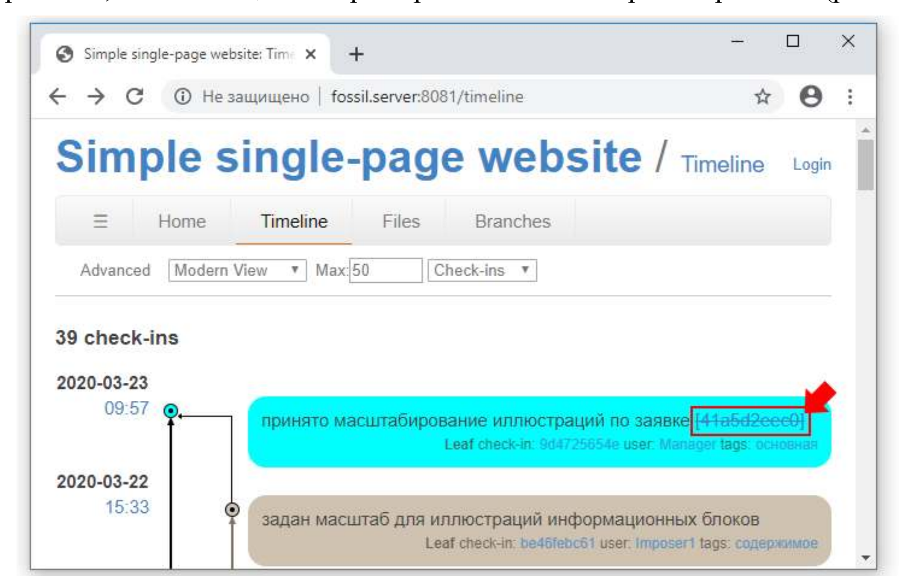


Рис. 5.15

Переключение с описания точки сохранения на карточку заявки

Кроме того, она попала в список точек сохранения, ассоциированных с заявкой на доработку, который можно посмотреть с помощью пункта Check-ins горизонтального меню карточки заявки (рис. 5.16).

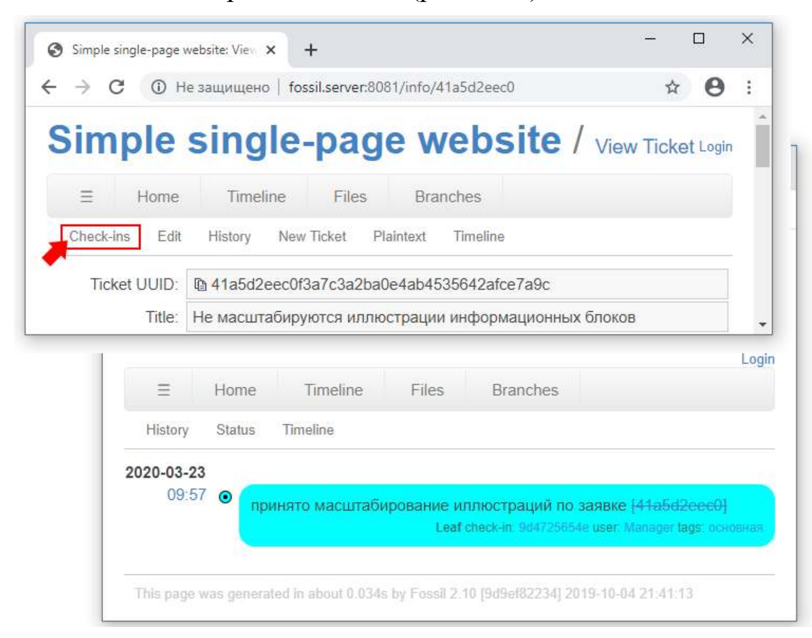


Рис. 5.16

Просмотр точек сохранения, относящихся к заявке на доработку

В рассмотренном примере заявка на доработку последовательно побывала в состояниях: «создана», «на рассмотрении», «выполнена», «проверена» и, наконец, «закрыта». Проследить всю историю этих изменений можно на шкале времени, которая открывается с помощью пункта Timeline горизонтального меню, присутствующего на карточке заявки (рис. 5.17). Идентификатор заявки зачёркнут, потому что она находится в состоянии «закрыта».

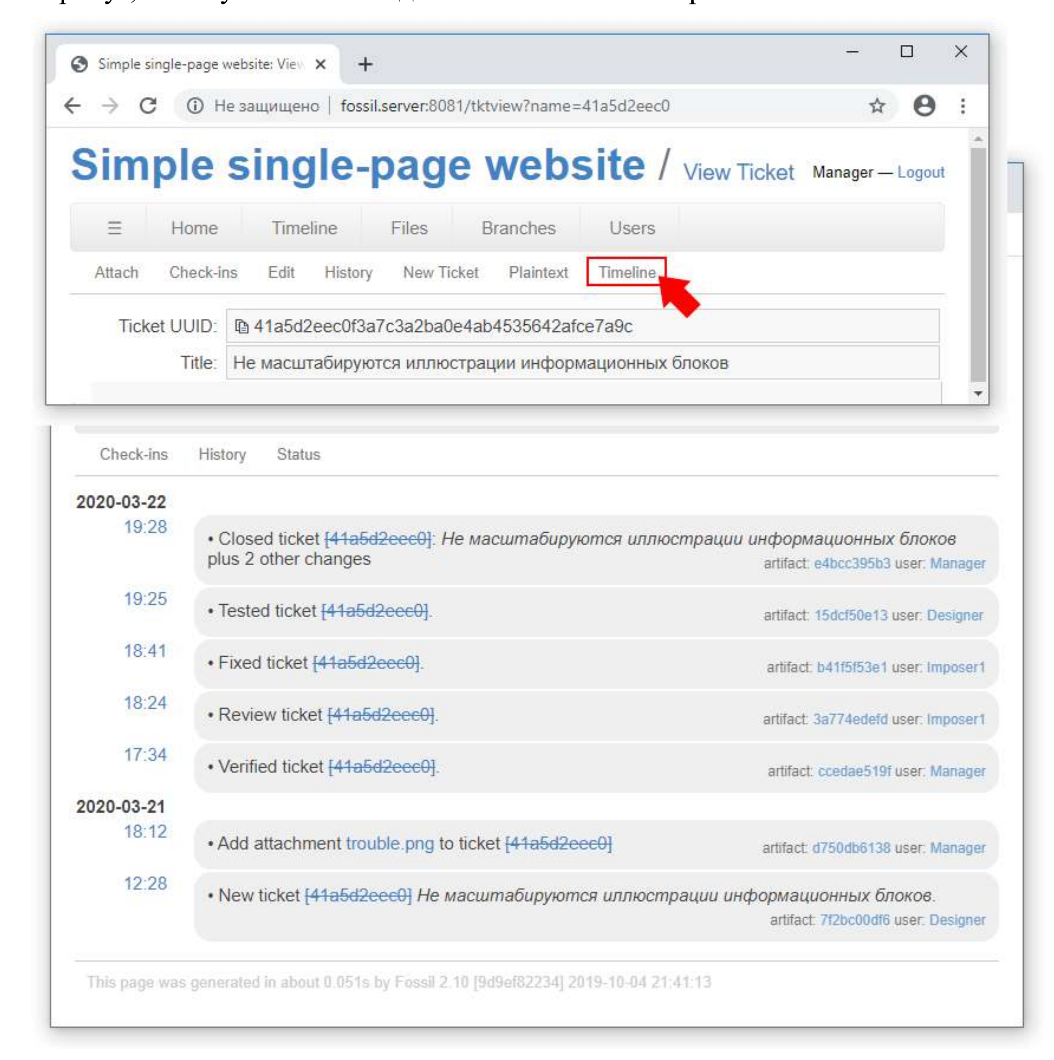


Рис. 5.17

Шкала времени заявки на доработку

Эти же сведения можно получить с помощью команды:

```
fossil ticket history 41a5d2eec0
```

По мере смены состояний менялся соответствующий заявке цвет строки в таблице заявок (рис. 5.18). При активной работе над проектом, когда в системе

зарегистрировано одновременно множество заявок, это помогает быстро оценить общую картину.

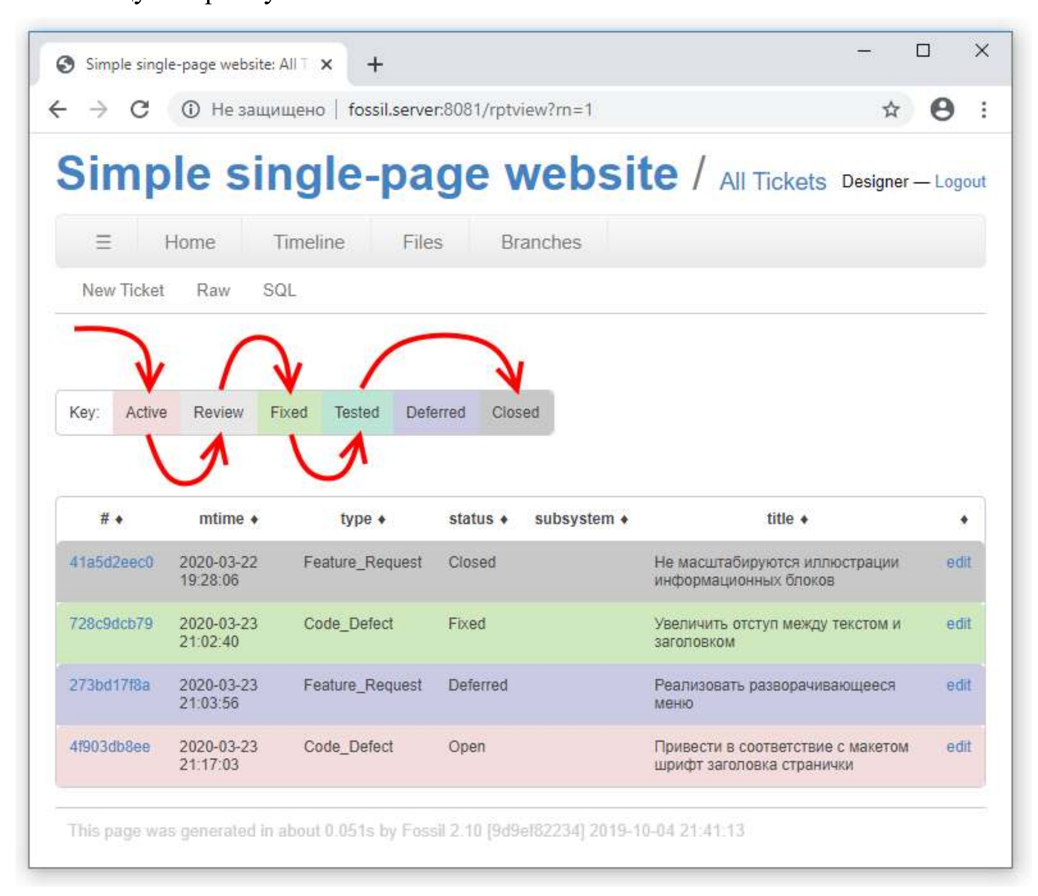


Рис. 5.18

Цветовое кодирование этапов обработки заявок

Строки таблицы можно сортировать по значениям, находящимся в колонках, если щёлкнуть по ромбу, изображённому справа от заголовка соответствующей колонки. В результате сортировки ромб заменится на изображение стрелки вверх или вниз в зависимости от установившегося порядка.

Глава 6
СИСТЕМА ОПОВЕЩЕНИЙ

6.1. Уведомления о событиях

После ознакомления с порядком взаимодействия участников проекта при обслуживании заявок на доработку может возникнуть вопрос: а как пользователи будут узнавать об изменении состояния заявки, которая их интересует? В дальнейшем будут рассмотрены другие подсистемы Fossil, в частности — система документирования и форум, которым тоже придётся уделять внимание. Кроме того, нужно постоянно быть в курсе последних событий, происходящих в ядре системы Fossil — репозитории разрабатываемого проекта.

Ответить на поставленный вопрос можно в административном стиле — рекомендовать участникам проекта периодически обращаться к веб-интерфейсу этих подсистем, чтобы узнавать о происходящих в них событиях. Однако это приведёт к непроизводительным затратам времени, снижению продуктивности разработчиков и может даже вызвать сомнения в целесообразности использования всех перечисленных инструментов. Поэтому хотелось бы найти более удобное решение для распространения новостей.

В системе Fossil реализован механизм, который позволяет оперативно информировать участников проекта о происходящих в ней событиях. Уведомления о новостях формируются в формате сообщений электронной почты, который подробно описан в документе RFC 822. Из системы Fossil эти уведомления могут передаваться во внешнюю среду одним из следующих способов:

- на стандартный вход указанной в настройках утилиты (как правило, программы sendmail);
- путём записи в базу данных SQLite;
- выгружать заданный каталог в виде отдельных файлов;
- отправляться на почтовый сервер по протоколу SMTP;
- передаваться по протоколу SMTP непосредственно адресатам.

Участники проекта наверняка общаются между собой по электронной почте, поэтому вариант передачи уведомлений Fossil через почтовый сервер выглядит наиболее привлекательным. Но реальность такова, что популярные публичные сервисы электронной почты вынуждены устанавливать барьеры на возможность использования их в качестве ретрансляторов в целях борьбы со спамом — потоком нежелательной корреспонденции. В настоящее время система Fossil не поддерживает авторизацию при работе по протоколу SMTP, что ещё больше снижает её шансы на использование совместно с внешними системами. Другими словами, отправлять письма напрямую на почтовые адреса в доменах mail.ru или gmail.com не получится.

Но если электронная почта разработчиков обслуживается их собственным внутренним сервером, то он может быть сконфигурирован с более мягкими требованиями к безопасности, что позволит ему принимать подключения от серве-

ра Fossil по протоколу SMTP. В этом случае его защита от несанкционированного использования должна быть реализована другими средствами, например с помощью брандмауэра. Ввиду разнообразия программных продуктов, которые могут использоваться для построения почтовой системы организации, настройка самого почтового сервера здесь не освещается. Она должна производиться в соответствии с документацией на почтовый сервер.

Ниже рассматривается настройка системы Fossil для рассылки оповещений через внутренний почтовый сервер коллектива разработчиков по протоколу SMTP.

6.2. Настройка рассылки оповещений

Если решено внедрить информирование по электронной почте с использованием протокола SMTP, то первое, что надо сделать, — это проверить, сможет ли сервер Fossil взаимодействовать с почтовым сервером. Для этого на сервере Fossil надо ввести команду:

```
fossil test-smtp-probe --direct mail.server noreply@fossil.server
```

В приведенной команде mail.server — это сетевое имя почтового сервера (вместо него можно указать IP-адрес). В случае успешного прохождения проверки на экране должны появиться строки информационного обмена с почтовым сервером по протоколу SMTP:

```
Connection to "mail.server" S: 220 192.168.56.1 ESMTP SubEthaSMTP null
C: EHLO noreply@fossil.server
S: 250-192.168.56.1
S: 250-8BITMIME
S: 250-AUTH LOGIN
S: 250 Ok
C: QUIT
S: 221 Bye
```

Возможно, вместо этого через некоторое время после запуска команды на экране появится единственная строка:

```
cannot connect to host mail.server:25
```

Это означает, что Fossil не смог подключиться к TCP-порту номер 25 почтового сервера mail.server. Возможно, дело в настройках брандмауэра одного или обоих серверов. В любом случае, речь идёт о невозможности сетевого подключения.

Может получиться так, что после запуска команды на экране появятся следующие сообщения:

```
Connection to "mail.server"
S:
S:
Error: timeout
```

К сожалению, это свидетельствует о неспособности используемой версии Fossil взаимодействовать с почтовым сервером по протоколу SMTP. Такое пока

что происходит на операционных системах Windows и Minix. Скорее всего, эта проблема будет в скором времени решена, возможно, даже с помощью читателей этой книги. А пока что можно констатировать, что Fossil 2.8 в операционной системе Mageia GNU/Linux 7.1 с почтовым сервером работает правильно. Вероятно, не возникнет проблем и на других версиях Linux.

Будем считать, что проверка подключения к SMTP-серверу завершилась успешно. Тогда можно перейти к настройке сервера Fossil. Для этого надо зайти в веб-интерфейс Fossil и авторизоваться пользователем с правами владельца репозитория (прав администратора недостаточно). Затем следует выбрать пункт меню Admin | Notification (рис. 6.1).

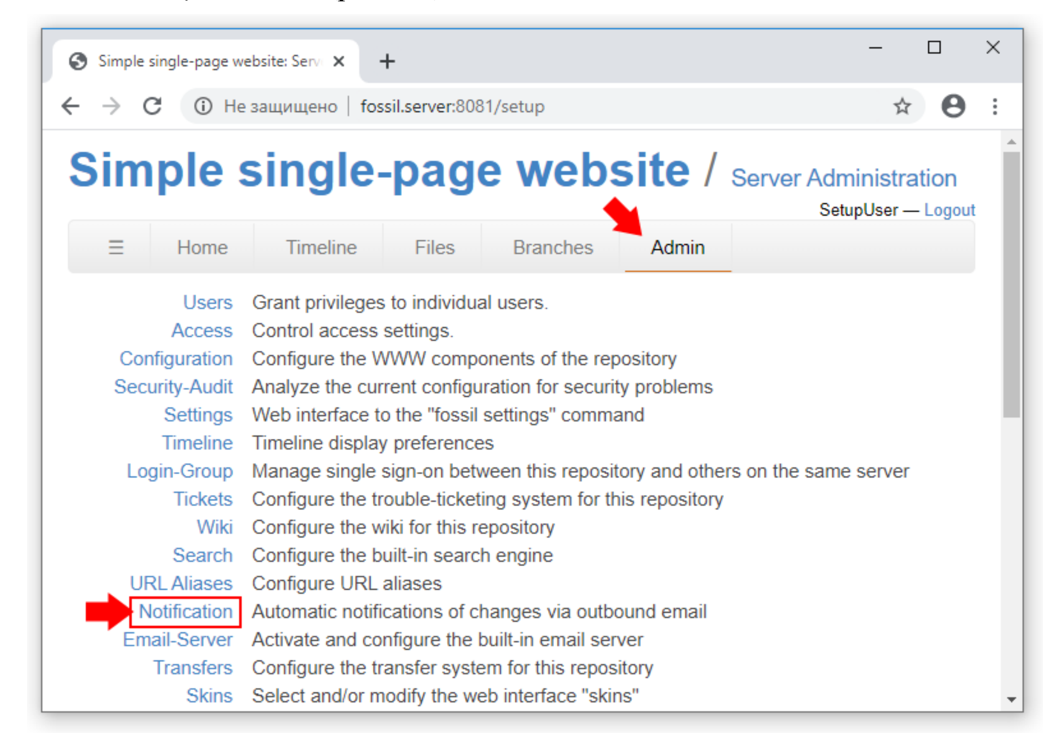


Рис. 6.1

Переход к форме настройки оповещений по электронной почте

Откроется форма настройки оповещений по электронной почте, в которой надо заполнить следующие поля (рис. 6.2):

- Canonical Server URL (базовый адрес сервера для использования в URL) — адрес сервера Fossil, с которого будут начинаться ссылки в оповещениях. Например, если указать в этом поле значение http://fossil.server:8081, то в оповещении о заявке с идентификатором 41a5d2eec0 ссылка на заявку будет записана в виде http://fossil.server:8081/tktview?name=41a5d2eec0;
- "Return-Path" email address — обратный адрес для указания в уведомлениях;
- Repository Nickname (псевдоним репозитория) — короткое название репозитория для указания в теме письма, например ssp-website;

• Email Send Method (способ отправки электронной почты) — для рассматриваемого примера, когда сообщение подлежит пересылке на SMTP-сервер, надо выбрать из списка значение SMTP relay. Но возможны и другие варианты:

– Pipe to a command — сообщение передаётся на стандартный вход указанной в отдельном поле утилиты командной строки (обычно используется sendmail);

– Store in a database — сообщения записываются в указанную в отдельном поле базу данных SQLite;

– Store in a directory — сообщения записываются в файлы по указанному в отдельном поле пути;

• SMTP relay host — адрес сервера электронной почты. Например, mail.server. По умолчанию будет использоваться стандартный TCP-порт для протокола SMTP номер 25, но при необходимости номер порта можно явно задать через двоеточие после имени сервера;

• Administrator email address — адрес электронной почты администратора сервера, на который будут отправляться сообщения о технических проблемах.

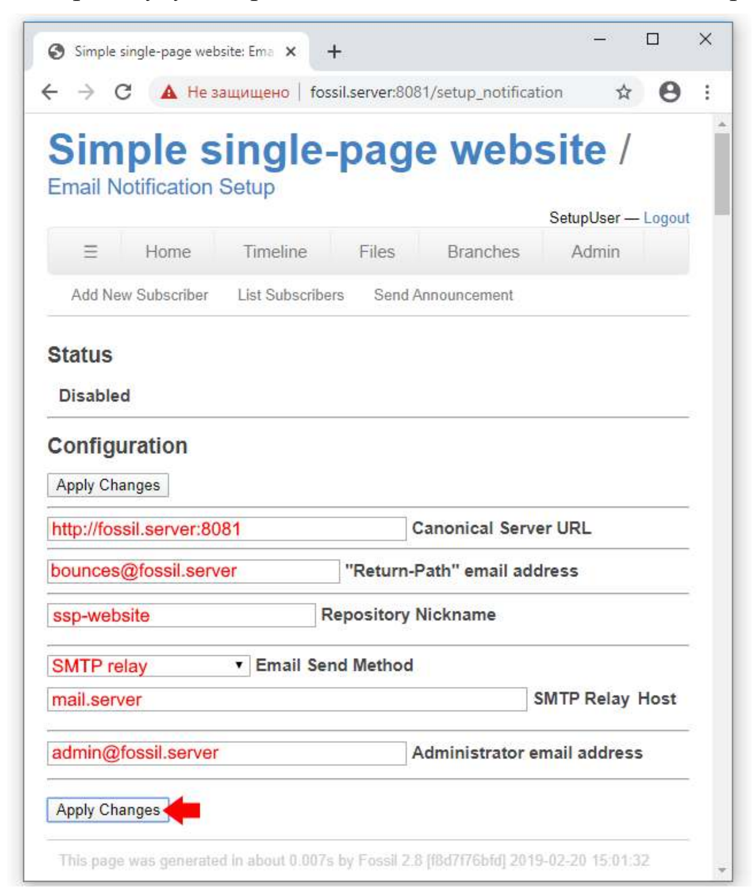


Рис. 6.2

Настройка оповещений по электронной почте

После заполнения всех полей надо нажать кнопку Apply Changes, которая продублирована вверху и внизу формы, дважды. После этого в заголовке формы появится текст: Status Outgoing Email: Relay to mail.server using SMTP (Состояние исходящей почты: Пересылка на сервер mail.server с использованием SMTP).

Посмотреть актуальные настройки системы рассылки оповещений можно с помощью команды:

```
fossil alerts settings -R sspwebsite.fossil
```

Поскольку эти настройки выполняются и хранятся только в сетевом репозитории, команду нужно выполнять непосредственно на сервере Fossil. С помощью параметра -*R* в команде указывается путь к файлу сетевого репозитория. После выполнения настроек через веб-интерфейс, как было только что рассмотрено, команда выведет на экран следующие сведения:

```
email-self (local) bounces@fossil.server
email-send-command
email-send-db
email-send-dir
email-send-method (local) relay
email-send-relayhost (local) mail.server
```

При необходимости значения выведенных полей можно изменить из командной строки, но тоже только на сервере. Например, если почтовый сервер ожидает подключений на TCP-порту номер 2525, то можно ввести команду:

```
fossil alerts settings email-send-relayhost mail.server:2525 -R
sspwebsite.fossil
```

Теперь можно проверить, как работает система оповещений по электронной почте, отправив тестовое сообщение одному из участников проекта. Пусть это будет дизайнер. Для этого прямо на форме настройки оповещений в дополнительном горизонтальном меню надо воспользоваться пунктом Send Announcement, после чего заполнить конверт письма и нажать кнопку Send Message (рис. 6.3).

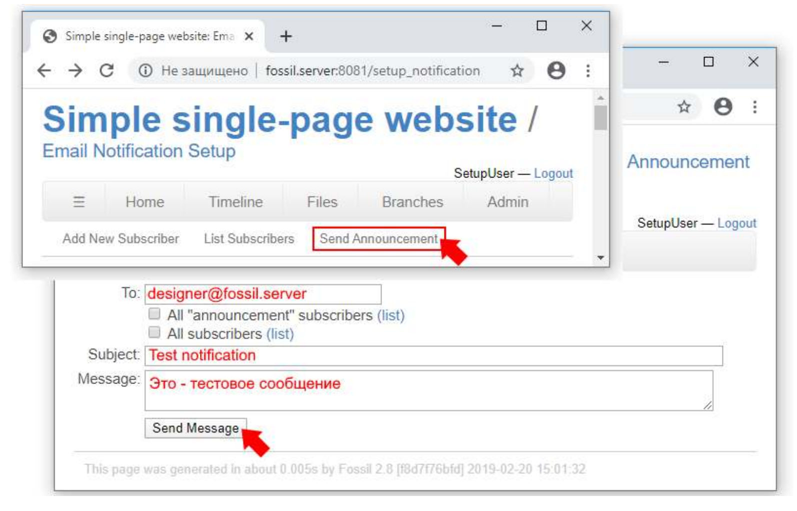


Рис. 6.3

Отправка тестового сообщения

Если система работает правильно, то через несколько секунд дизайнер сможет прочитать отправленное ему сообщение (рис. 6.4).

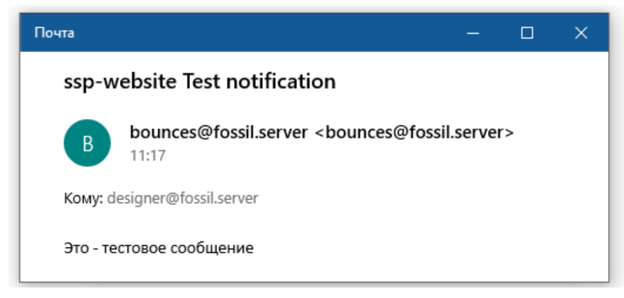


Рис. 6.4

Полученное тестовое сообщение

6.3. Подписка на оповещения

Итак, система оповещений по электронной почте настроена и проверена. Но чтобы получать такие оповещения, участники проекта должны попасть в число подписчиков. Сделать это можно двумя способами.

Во-первых, сделать пользователя Fossil подписчиком может владелец или администратор репозитория. Если первому достаточно воспользоваться пунктом меню Add New Subscriber (Добавить нового подписчика) на недавно рассмотренной форме настройки системы рассылки, то администратору потребуется перейти на форму настройки почтовых рассылок посредством пункта раскрывающегося меню Logout | Email Alerts или путём ввода в адресной строке веб-браузера: http://fossil.server:8081/subscribe (рис. 6.5).

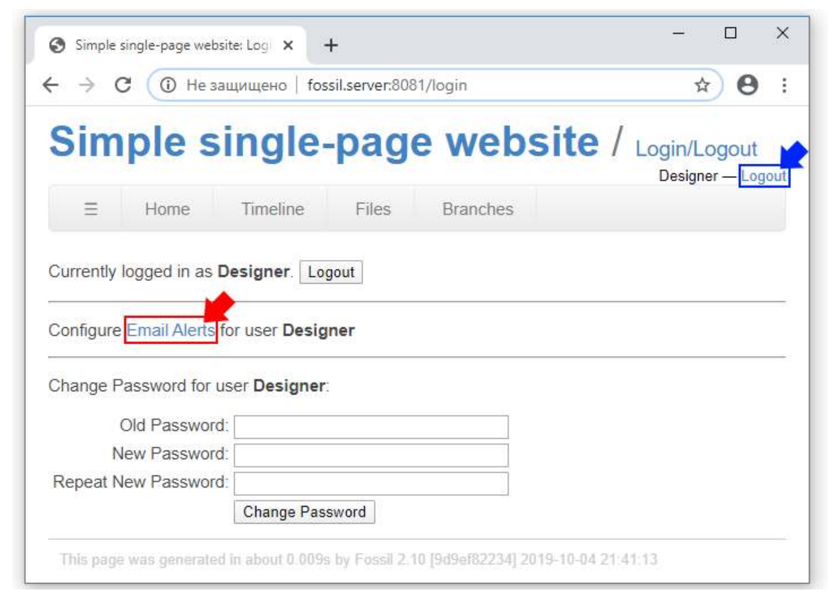


Рис. 6.5

Переход к настройке почтовых рассылок

В форме добавления подписчика понадобится заполнить поля Email Address: (Адрес электронной почты) и User: (Пользователь) (рис. 6.6). Значения в этих полях не обязательно должны соответствовать указанным в учётных записях пользователей Fossil. Однако если указать имя несуществующего пользователя, то он не сможет пройти проверку адреса электронной почты, о которой говорится ниже.

Затем надо отметить категории сущностей Fossil, о которых должны отправляться уведомления:

- Announcements — анонсы;
- Check-ins — точки сохранения в репозитории;
- Forum Posts — сообщения на форуме;
- Ticket changes — изменения заявок на доработку;
- Wiki — документация.

Далее надо выбрать один из двух способов информирования: Individual Emails (Индивидуальные письма) или Daily Digest (Ежедневная рассылка).

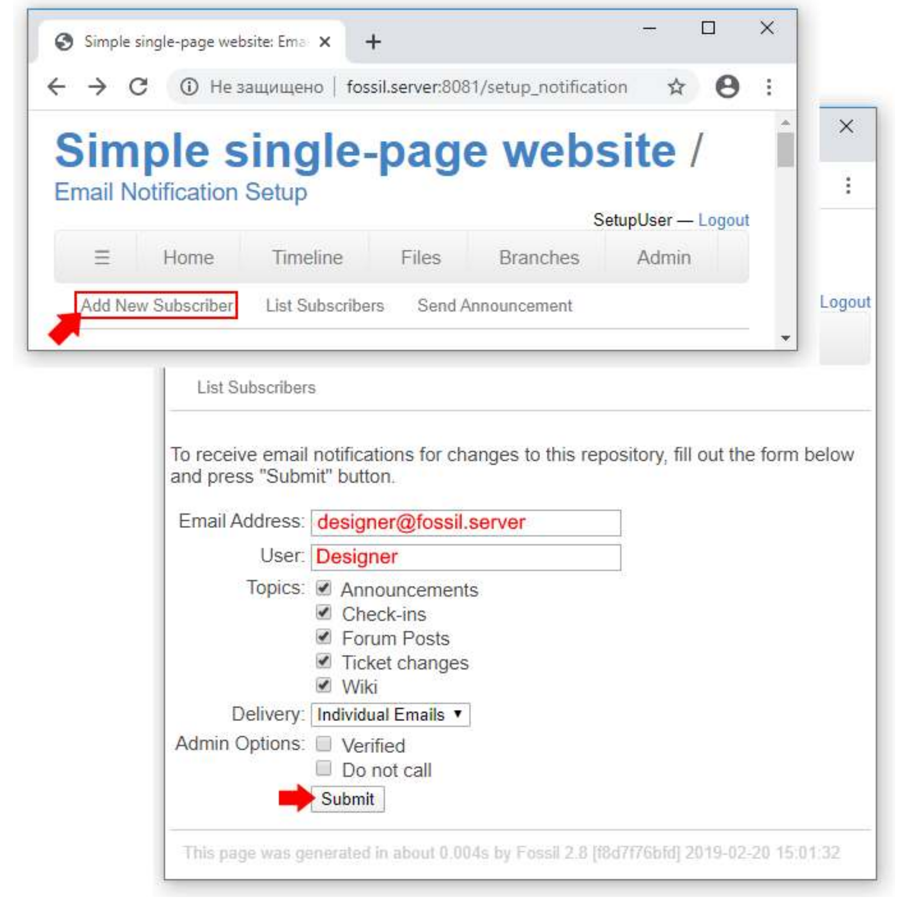


Рис. 6.6

Добавление пользователя в перечень подписчиков

Кроме этого, на форме присутствуют ещё два флажка в группе Admin Options: (Опции администрирования): Verified (Проверен) и Do not call (Не рассылать). Первый флажок в итоге будет установлен независимо от того, как была заполнена форма. А установка второго флажка означает, что рассылка этому пользователю не должна отправляться до того момента, пока этот флажок не будет снят.

По завершении заполнения формы надо нажать кнопку Submit (Подтвердить). На экране отобразится форма редактирования учётной записи только что созданного подписчика. На этом этапе подписчик уже добавлен в список, в чём можно убедиться, воспользовавшись пунктом дополнительного меню List Subscribers.

Единственное, на чём стоит задержать внимание, — уже упомянутый флажок Verified. На открывшейся форме он будет установлен, и сейчас администратору предоставляется возможность его снять. Если флажок отсутствует, то будет считаться, что адрес электронной почты, указанный в форме подписки, нуждается в проверке. Механизм проверки заключается в том, что рассылка на такой адрес не будет производиться до тех пор, пока пользователь, которому назначена подписка, не авторизуется в системе Fossil и не зайдёт на страницу настройки своей подписки. Лишь после этого учётная запись подписчика активируется, о чём ему сообщит надпись: Your email alert subscription has been verified! (Ваша подписка на уведомления прошла проверку!).

Теперь должно быть понятно, что если у пользователя нет учётной записи Fossil или разрешения на самостоятельное управление подписками, о котором будет сказано ниже, то он не сможет подтвердить подписку, оформленную на него администратором. Поэтому для таких пользователей администратор не должен убирать флажок Verified, если он не хочет воспрепятствовать получению ими уведомлений.

Преимущество создания подписчика владельцем или администратором репозитория Fossil заключается в том, что таким способом подписку на уведомления можно оформить не только на участника проекта, но и на любого другого человека, имеющего учётную запись на сервере электронной почты. Например, это может быть директор организации, который не участвует напрямую в работе над проектом, но желает отслеживать через получение рассылок активность своих сотрудников.

Второй способ оформления подписки на уведомления Fossil — участником проекта самостоятельно. Он доступен только в том случае, если в настройках учётной записи пользователя администратором установлена отметка Email Alerts (Почтовые уведомления) (рис. 6.7).

Пользователи, имеющие разрешение на управление своими подписками, после авторизации в Fossil получают возможность произвести их настройку, перейдя по гиперссылке Email Alerts с экрана выхода из системы Logout (рис. 6.8) или по ссылке http://fossil.server:8081/alerts.

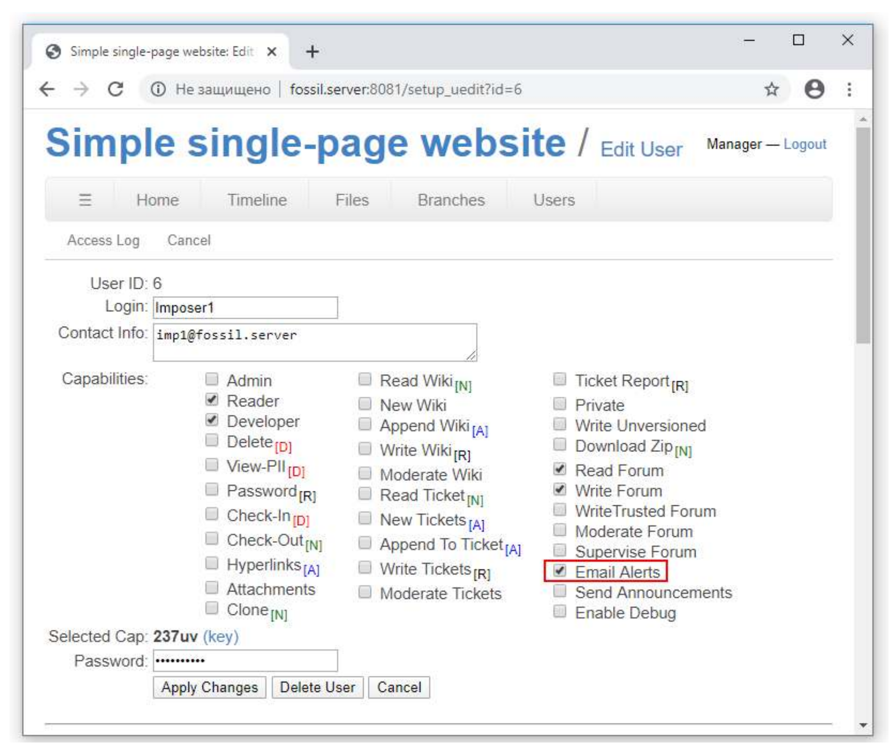


Рис. 6.7

Разрешение на самостоятельное управление подписками

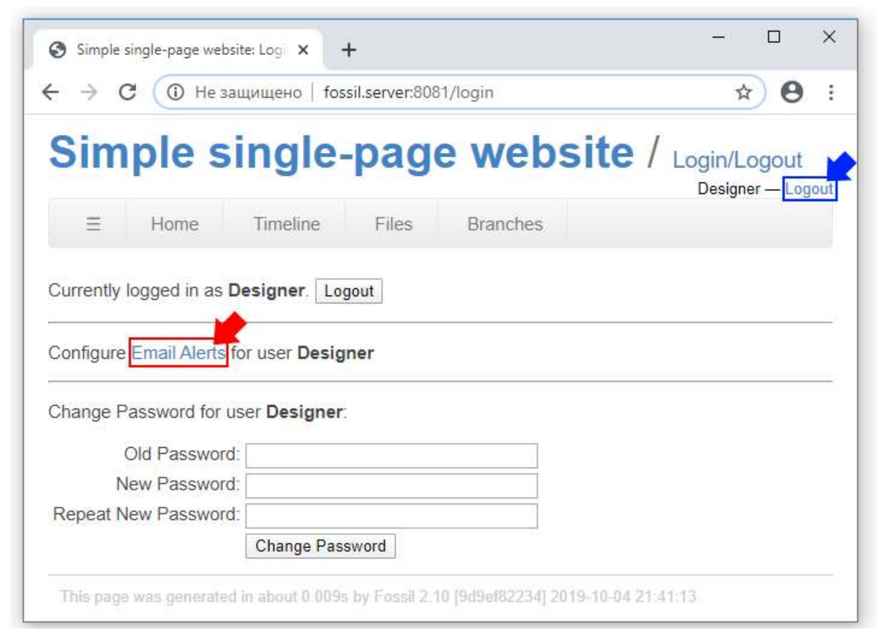


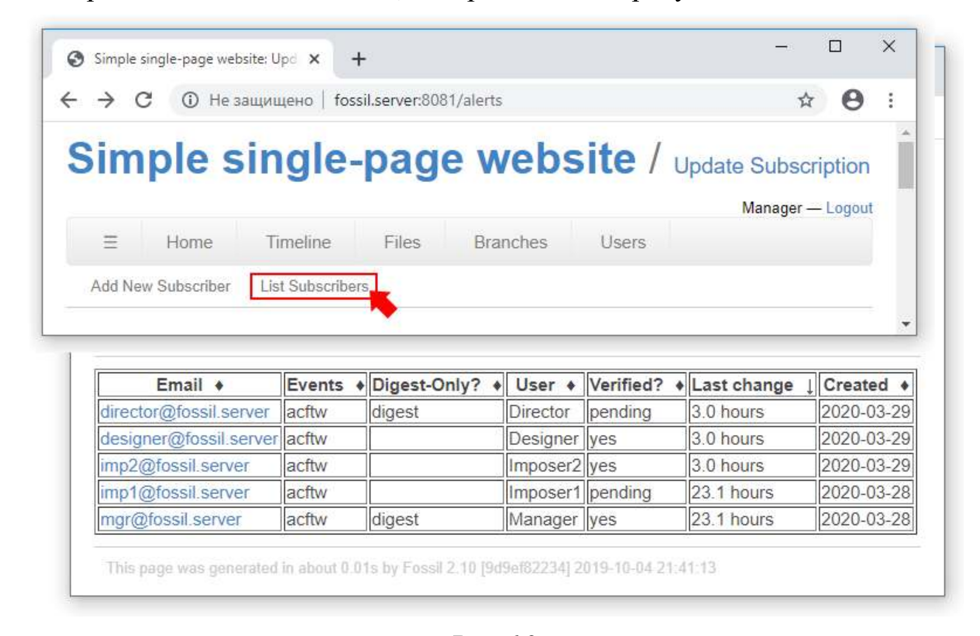
Рис. 6.8

Ссылка на настройку почтовых рассылок

Форма, которая предлагается к заполнению пользователю, является сокращённым аналогом рассмотренной выше формы, заполняемой администратором Fossil. На ней имеется только поле Email Address: для указания адреса электронной почты, куда должны отправляться сообщения, группа флажков Topics: с возможностью выбора категорий сообщений и поле со списком Delivery:, позволяющее указать способ отправки сообщений (отдельными письмами или ежедневным отчётом). Учётная запись подписчика, который подписался самостоятельно, сразу становится активной.

Если у пользователя имеется разрешение на самостоятельное управление подписками, то, независимо от того, был он подписан на уведомления Fossil администратором или подписался на них самостоятельно, он может отказаться от подписки. Для этого на форме управления подписками надо нажать кнопку Unsubscribe (Отписаться), потом установить флажок в появившемся поле Verify: Unsubscribe (Проверка: отписаться) и ещё раз нажать кнопку Unsubscribe.

Каким бы способом ни были оформлены подписки на оповещения, все подписчики попадают в единый список, который доступен администратору Fossil через пункт меню List Subscribers или по ссылке http://fossil.server:8081/subscribers. Он отображается в виде таблицы, изображённой на рисунке 6.9.



Email ♦	Events ♦	Digest-Only? ♦	User ♦	Verified? ♦	Last change ↓	Created ♦
director@fossil.server	acftw	digest	Director	pending	3.0 hours	2020-03-29
designer@fossil.server	acftw		Designer	yes	3.0 hours	2020-03-29
imp2@fossil.server	acftw		Imposer2	yes	3.0 hours	2020-03-29
imp1@fossil.server	acftw		Imposer1	pending	23.1 hours	2020-03-28
mgr@fossil.server	acftw	digest	Manager	yes	23.1 hours	2020-03-28

Рис. 6.9

Перечень подписчиков

В первой колонке Email таблицы отображаются адреса электронной почты, на которые оформлены подписки. Они являются гиперссылками, при переходе по которым открывается форма для редактирования соответствующей подписки.

Во второй колонке Events в закодированном виде отображаются наборы событий, на которые генерируются оповещения. Каждому событию соответствует буква латинского алфавита:

- (a)nnounce — анонсы;
- (c)heck-in — точки сохранения в репозитории;
- (f)orum — сообщения на форуме;
- (t)icket — заявки на доработку;
- (w)iki — страницы документации.

В третьей колонке Digest-Only отображаются способы доставки оповещений: индивидуальными письмами (пустая ячейка таблицы) или ежедневными подборками (digest).

В четвёртой колонке User отображаются имена пользователей Fossil, на которых оформлена подписка.

В пятой колонке Verified отображаются признаки того, что адрес электронной почты подтверждён (verified) или ещё находится в ожидании подтверждения (pending).

В шестой колонке Last Change и в седьмой колонке Created отображается информация о времени последней модификации и дате создания подписки соответственно.

6.4. Система в действии

Рассылкой сообщений занимается сервер Fossil. События, о которых следует уведомить подписчиков, накапливаются в базе данных и находятся там в ожидании обработки.

Обработка ожидающих событий производится подсистемой backoffice, которая активируется запросами, поступающими от пользователей Fossil. Например, когда кто-то записывает моментальный снимок рабочего каталога в репозиторий, одновременно с созданием точки сохранения запускается процесс, обслуживающий фоновые задачи. В число таких задач входит и рассылка уведомлений.

Такой принцип выполнения фоновых задач не требует наличия постоянно активного серверного процесса, однако для его нормального функционирования запросы в систему Fossil должны поступать достаточно регулярно. Если никаких действий с репозиторием не происходит, то не будут выполняться фоновые задачи, а значит, задержится и обработка ожидающих событий с рассылкой уведомлений.

Администратор может посмотреть состояние системы оповещений через пункт раскрывающегося меню Administration pages | Stats. Помимо подробной информации о репозитории в нижней части открывшегося окна находятся поля, имеющие непосредственное отношение к рассматриваемому вопросу (рис. 6.10):

- Backoffice — время последнего запуска фоновых процессов;
- Outgoing Email (Исходящая почта) — способ осуществления рассылки уведомлений;

• Pending Alerts (Ожидающие уведомления) — количество записей о событиях, которые ожидают обработки, отдельно — индивидуальными письмами (normal) и отдельно — ежедневными подборками (digest);

• Subscribers (Подписчики) — количество подписчиков, отдельно — участвующих в получении уведомлений (active), и отдельно — всего, вместе с теми, у кого отключено получение оповещений (total).

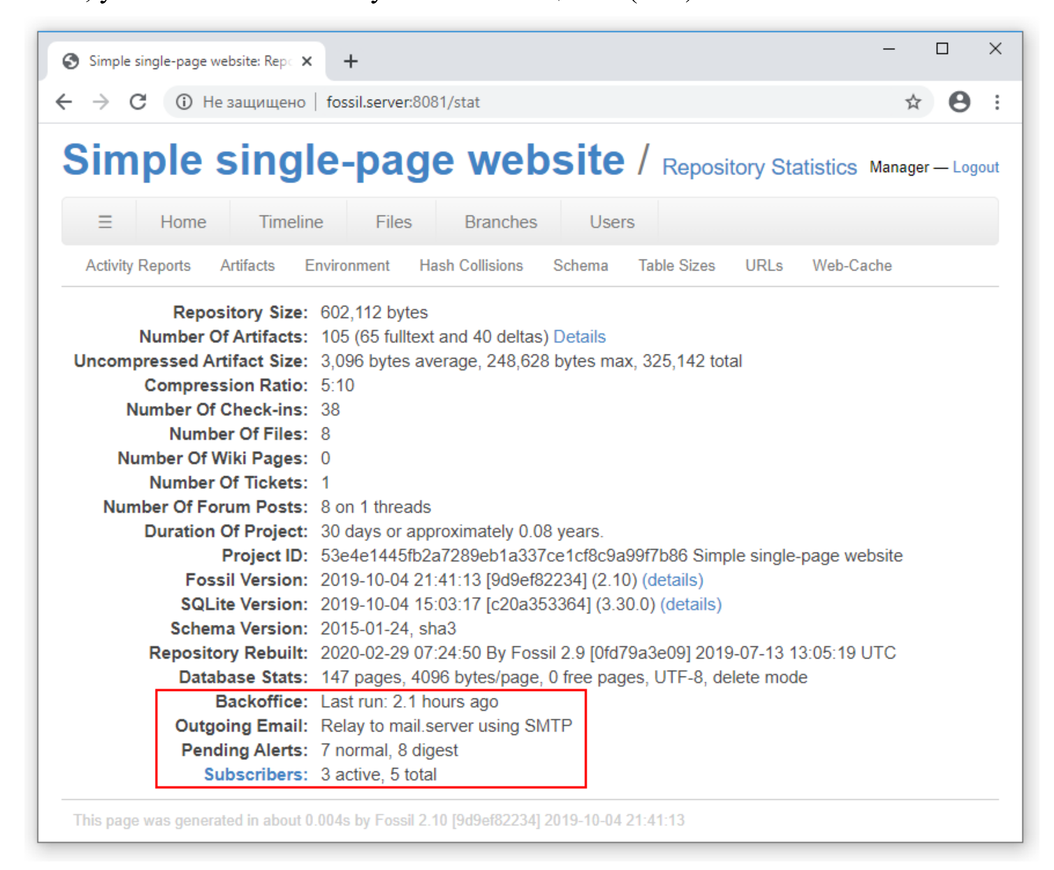


Рис. 6.10

Просмотр состояния системы оповещений через веб-интерфейс

Эти же сведения можно получить с помощью запроса, выполненного к файлу сетевого репозитория:

```
fossil alerts status -R sspwebsite.fossil
```

Приведенная команда выведет информацию о настройках системы рассылки уведомлений и её текущем состоянии:

```
email-self (local) bounces@fossil.server
email-send-command
email-send-db
email-send-dir
email-send-method (local) relay
```

```
email-send-relayhost (local) mail.server
pending-alerts 7
pending-digest-alerts 8
total-subscribers 5
active-subscribers 3
```

В рассматриваемом примере в системе рассылки накопилось 15 уведомлений: семь для отправки индивидуальными сообщениями и восемь — для ежедневной рассылки. Чтобы не дожидаться, когда их отправит фоновый процесс, можно дать команду:

```
fossil alerts send -R sspwebsite.fossil
```

Эта команда отправит только индивидуальные сообщения. Для отправки писем с ежедневными рассылками в неё надо добавить параметр *--digest*.

Если репозиторий используется не очень интенсивно, приведенную команду можно задействовать в автоматически выполняемых сценариях для регулярной рассылки уведомлений.

Глава 7
СИСТЕМА ДОКУМЕНТИРОВАНИЯ

7.1. Программные документы

Документация является неотъемлемой частью программных проектов. Важность этого компонента подчёркивается наличием отдельнй группы стандартов, посвящённых документации, — ГОСТ 19 ЕСПД (Единая система программной документации). Многие стандарты этой группы были разработаны ещё в 1970-х гг. и в своём большинстве по сегодняшний день не утратили актуальности. В соответствии со стандартом к программным относят документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ.

Почему документация, или всестороннее техническое описание программы, так важна? Документально зафиксированные требования исключают неоднозначности и вариации в их трактовке, описания различных аспектов реализации и функционирования упрощают последующую модификацию, предотвращают ошибки на этапах внедрения и эксплуатации.

С перечнем рекомендуемых стандартом сопроводительных документов можно ознакомиться в ГОСТ 19.101-77, который описывает целых 16 их видов. Это — исчерпывающий список, и не каждая разработка нуждается в полном наборе документов. В то же время каждый этап разработки программного продукта должен документироваться.

В жизненном цикле компьютерной программы можно выделить четыре основные фазы:

- проектирование;
- реализация;
- внедрение;
- эксплуатация.

Проектирование начинается с составления технического задания, в котором описываются цели проекта и основные требования к разрабатываемому продукту. На основе технического задания разрабатывается детальная спецификация, включающая перечень компонентов, которые необходимо реализовать, и описание способа их взаимодействия в создаваемой системе. Желательно сразу описать и методику испытаний разрабатываемого продукта, которая позволит понять, что поставленные перед проектом цели достигнуты и программа соответствует предъявляемым к ней требованиям.

На стадии проектирования документы создаются при активном участии как разработчика, так и заказчика, а результат их совместной работы должен быть утверждён обеими сторонами. Важно, чтобы эти документы были максимально конкретными. Не следует пытаться обойтись в них общими фразами и сглаживать острые углы. Детальные перечни функций, подробное описание вариантов использования, конкретные значения основных параметров уменьшат

количество вопросов на этапе реализации и исключат конфликты на этапе приёмки готового продукта.

На основе технического задания и спецификации выполняется разработка программы — наступает этап реализации. Основным документом этого этапа станет текст программы на языке программирования. Однако, несмотря на его исчерпывающее содержание, опытные программисты подтвердят, что пользоваться им без вспомогательных материалов не всегда легко. Следование соглашениям по оформлению текстов программ может сгладить проблему, но не решает её полностью. Большим подспорьем к пониманию программы являются внедрённые в её текст комментарии.

Не следует забывать, что текст программы — это ещё далеко не сама программа. А процедура превращения исходных текстов в исполняемый код может оказаться нетривиальной задачей. Поэтому необходимо подробно описать процесс сборки: какие инструменты, с какими параметрами и в какой последовательности для этого применяются, как выполняется проверка правильности программы, получившейся в результате этого процесса.

Одновременно с написанием текста программы полезно фиксировать описания реализуемых алгоритмов, объяснять, почему было выбрано то, а не иное решение. На этом же этапе надо разрабатывать и документы для тех, кто будет готовить создаваемую систему к эксплуатации, и тех, кто ею будет пользоваться. Тогда к моменту наступления очередных этапов — внедрения и эксплуатации — информационное обеспечение в виде руководств оператора и пользователя будет уже подготовлено. По большому счёту, эту книгу можно считать руководством пользователя и руководством оператора системы Fossil.

На практике программа используется не бесконечно, а до определённого момента — пока не утратят актуальности выполняемые ею функции и пока будет существовать возможность её адаптации к изменяющимся внешним условиям. Этот момент обычно не показывается на формальной схеме жизненного цикла, потому что его наступление не всегда можно предсказать, а тем более нелегко описать действия, которые должны ему сопутствовать.

Как правило, выводимую из эксплуатации программу сменяет другая. Какое-то время в эксплуатации могут находиться обе системы — старая и новая (для новой системы этот этап называется опытно-промышленной эксплуатацией). Такой подход позволяет сравнить результаты работы двух систем и при необходимости внести исправления в новую систему. Обычно при переходе выполняется перенос данных из старой системы в новую. В любом случае надо выполнить архивирование данных, накопленных в старой системе, и организовать их хранение вместе с программным окружением, чтобы при необходимости можно было получить к ним доступ.

Программную документацию можно разделить на два класса — формальную и неформальную. Формальная документация должна быть разработана и включена в комплект поставки программного продукта независимо от желания разработчиков. Те документы, которые перечислены в стандартах, являются частями формальной документации.

Но для накопления и сохранения знаний о разрабатываемой системе, которые зачастую возникают непосредственно в ходе работы над проектом, разработчики ведут собственные записи. Это могут быть заметки и наброски в тетради, ежедневнике или даже на отдельных листках. Такая информация относится к неформальной документации. Несмотря на то, что она не всегда хорошо структурирована и оформлена, содержащиеся в ней сведения могут составлять высокую ценность для проекта.

Подсистема Fossil Wiki представляет собой набор взаимосвязанных веб-страниц, хранящихся в репозитории вместе с файлами разрабатываемого проекта, и хорошо подходит для ведения неформальной документации.

7.2. Оформление текстов

Текст плохо читается, если его оформлению не было уделено должного внимания. Правила пунктуации русского языка описывают давно вошедшие в привычку обозначения, позволяющие разделять слова (пробелы), предложения (точки), выделять прямую речь (кавычки) и обозначать другие структурные элементы текста. Таким образом, пунктуация — это универсальный язык разметки, пригодный для использования как в рукописном, так и в печатном тексте.

В настоящее время практически на любом персональном компьютере установлен визуальный текстовый процессор типа Microsoft Word или LibreOffice Writer — если не для подготовки документов, то для их просмотра и печати. Программы этого типа помимо самого текстового содержания документа отображают и позволяют изменять его оформление — размер и толщину шрифта, цвет символов и фона, способ выравнивания абзацев, величины отступов и интервалов. Стили оформления применяются к тексту сразу в ходе редактирования, в результате чего он всегда выглядит нарядным.

Но текстовые процессоры устроены таким образом, что оформление накладывается на текст путём выделения участков текста и выбора для них необходимых свойств из палитры цветов или списка шрифтов. Результат выбора сохраняется вместе с текстовым содержанием в файле, структура которого больше напоминает базу данных, чем файл с простым текстом. Изменить стилевое оформление путём отыскания в этом файле необходимых элементов и их корректировки практически невозможно.

Простые текстовые редакторы отличаются от визуальных текстовых процессоров тем, что позволяют создавать файлы, содержащие только набираемые на клавиатуре символы. Этого достаточно, чтобы создавать тексты программ. Более того, в текстах программ недопустимо присутствие посторонней информации об оформлении текста. Как правило, параметры отображения устанавливаются в настройках текстовых редакторов и применяются ко всему редактируемому документу целиком.

Однако программисты предпочитают использовать не простые текстовые редакторы, а так называемые редакторы с подсветкой синтаксиса. Они отличаются тем, что умеют распознавать синтаксические конструкции языка програм-

мирования, такие как заголовки функций, комментарии и литеральные константы, и выделяют их из остального текста цветом или начертанием символов. Это позволяет лучше ориентироваться в тексте программы, легче находить и исправлять ошибки (рис. 7.1).

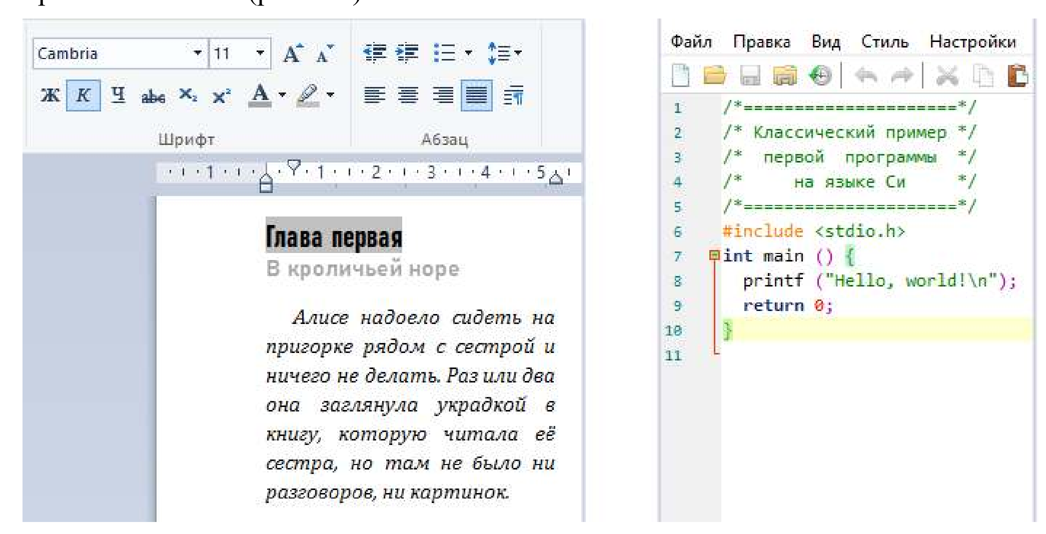


Рис. 7.1

Оформление текста в текстовом процессоре (слева) и текстовом редакторе с подсветкой синтаксиса (справа)

Языки разметки позволяют совместить преимущества текстовых процессоров и простых текстовых редакторов, чтобы получить для произвольного текста эффект, подобный тому, который делают для исходных текстов программ текстовые редакторы с подсветкой синтаксиса. С помощью языков разметки можно, используя только набираемые с клавиатуры символы, отметить смысловые блоки текста: заголовки, абзацы, перечисления, списки. Тогда любые программы, знакомые с языком разметки, смогут распознать структуру текста и по-разному оформить его структурные элементы.

Для внесения в произвольный текст информации о его смысловых блоках можно изобрести разные способы. Можно, например, заключать участки текста в своего рода скобки, как это делается с комментариями в языке Си: /* Это комментарий */. Можно использовать различную величину отступов от левого края. Важно лишь, чтобы принятое соглашение не входило в противоречие с остальным текстом, и желательно, чтобы оно было удобным для использования, потому что в отличие от текстовых процессоров, где оформление накладывается на текст инструментальным способом, писать на языке разметки зачастую приходится непосредственно человеку.

Рассмотрим языки разметки, которые предлагается использовать для оформления страниц документации в Fossil.

7.3. Язык разметки Wiki

Когда встречают слово Wiki (которое, кстати, имеет корни в гавайском языке, где оно означает «быстро»), то обычно представляют Википедию — грандиозный информационный ресурс, размещённый в сети Интернет. В системе Fossil Wiki — это всего лишь набор взаимосвязанных страничек с текстом, которые легко создавать через веб-интерфейс этой системы и которые удобно там же просматривать. А для того, чтобы документация лучше воспринималась, предлагается использовать специальный язык разметки.

Основным структурным элементом страницы текста является абзац. В разметке Wiki достаточно между абзацами текста вставлять пустые строки, чтобы система их правильно интерпретировала и отображала. Если требуется увеличить поле слева от абзаца, то составляющие его строки надо предварять двумя или более пробелами либо использовать табуляцию.

Довольно часто в текстах присутствуют перечисления каких-нибудь элементов, списки. Они бывают двух типов: нумерованные, в которых важен порядок, и поэтому каждый элемент снабжается своим номером, и ненумерованные, в которых порядок не важен. Чтобы оформить список без нумерации, надо каждый его элемент начинать с новой строки вида:

```
  *  очередной элемент
```

Важно, чтобы вокруг звёздочки находилось не менее двух пробелов с каждой стороны — тогда система отображения распознает в строке элемент ненумерованного списка и отобразит его со смещением от левого края, заменив звёздочку на закрашенный кружок. С учётом изложенных требований список можно оформить так:

```
  *  картина;
  *  корзина;
  *  картонка;
  *  собачонка.
```

Элементы нумерованного списка оформляются таким же образом, только вместо звёздочки ставится символ «решётка». Например, перечень этапов разработки проекта может быть записан в таком виде:

```
  #  Проектирование.
  #  Реализация.
  #  Внедрение.
  #  Эксплуатация.
```

Система отображения, встретив правильно оформленный блок элементов нумерованного списка, сама заменит решётки на последовательные числа (рис. 7.2).

Одно из наиболее значимых преимуществ электронной документации по сравнению с бумажным аналогом — возможность установления множественных связей между отдельными страницами. Не удивительно, что Wiki-разметка предоставляет удобное средство для создания гиперссылок. Чтобы сослаться из

какого-то места документа на другой документ, надо адрес ссылки заключить в квадратные скобки, например:

```
[http://fossil-scm.org]
```

Дама сдавала в багаж:

- картину;
- корзину;
- картонку;
- собачонку.

Этапы жизненного цикла компьютерной программы:

1. Проектирование.
2. Реализация.
3. Внедрение.
4. Эксплуатация.

Рис. 7.2

Отображение ненумерованного и нумерованного списков

Такая запись приведёт к тому, что в документе появится надпись «http://fossil-scm.org», которая одновременно является активной гиперссылкой на официальный веб-сайт проекта Fossil, т. е. при щелчке мышью по этой надписи в веб-браузере откроется сайт, соответствующий ссылке.

Если адрес веб-сайта длинный и его размещение в тексте нежелательно, то можно его замаскировать каким-нибудь текстом. В приведенном примере гиперссылку можно записать в таком виде:

```
[http://fossil-scm.org|официальный веб-сайт]
```

Теперь вместо адреса веб-сайта в документе будет присутствовать надпись «официальный веб-сайт», являющаяся гиперссылкой с указанным адресом.

Но наиболее значимое преимущество интегрированной в систему Fossil системы документирования заключается в том, что в качестве адреса для гиперссылки можно использовать идентификатор сущности системы контроля версий. Каждый объект, попавший в Fossil, получает свой уникальный номер — идентификатор, который выглядит, например, так: 01b66d3512ffbcb218ad136b10399850ac925eb0efc19f3236c2f3b98ca771fb. Если воспользоваться этим номером в качестве адреса гиперссылки (обычно достаточно указать его первые 10 символов), то система Fossil создаст гиперссылку на идентифицируемый им объект хранения, например моментальный снимок рабочего каталога в репозитории. Тогда при щелчке по гиперссылке [01b66d3512] откроется веб-страница с участком шкалы времени проекта, содержащая описание указанной точки сохранения.

Аналогичным образом можно ссылаться из документации на заявки на доработку, которые тоже имеют свои уникальные идентификаторы. Такой механизм позволяет быстро увязать все сведения, относящиеся к рассматриваемому на создаваемой страничке вопросу.

Если перечисленных средств форматирования недостаточно, то можно воспользоваться большинством тегов, составляющих язык гипертекстовой разметки HTML и позволяющих выделять в тексте отдельные элементы. Тег — это последовательность из одной или нескольких букв английского алфавита и цифр, размещённая между угловыми скобками < и >. Например, <hr> — тег, обозначающий горизонтальную разделительную линию в тексте.

Для обозначения участка текста обычно требуется указать его границы — места, где он начинается и заканчивается. Теги выступают в роли своеобразных многосимвольных скобок — открывающей, которая обозначает начало выделяемого участка текста, и закрывающей, которая обозначает его конец. Закрывающий тег отличается от открывающего тем, что в нём перед словом ставится символ /. Например, абзац обозначается так:

```
<P>здесь находится текст абзаца</P>.
```

Наиболее полезными для разметки текста элементами являются заголовки и таблицы. Заголовки текста оформляются с помощью тегов <H1>, <H2> ... <H7> (от *англ.* header — заголовок, цифра указывает на уровень заголовка — от крупного к более мелким). Например, начало сказки Льюиса Кэррола «Алиса в Стране чудес» может быть оформлено следующим образом:

```
<H1>Глава первая</H1>
<H2>В кроличьей норе</H2>
<P>Алисе надоело сидеть на пригорке рядом с сестрой и ничего не делать.</P>
```

Таблицы оформляются с помощью тегов <TABLE>, <TR>, <TH> и <TD> следующим образом:

```
<TABLE>
<TR><TH>N п/п</TH><TH>Название предмета</TH><TH>Учебных<BR>часов</TH></TR>
<TR><TD>1</TD><TD>Математический анализ</TD><TD>86</TD></TR>
<TR><TD>2</TD><TD>Аналитическая геометрия</TD><TD>72</TD></TR>
<TR><TD>3</TD><TD>Программирование</TD><TD>80</TD></TR>
</TABLE>
```

Тегом <TABLE> обозначаются начало и конец блока, в котором находится описание структуры таблицы. Таблица представляется последовательностью строк, которые заключены в теги <TR> (от *англ.* table row — строка таблицы). Каждая строка состоит из ячеек, для обозначения которых служат теги <TD> (от *англ.* table data — данные таблицы). В первой строке вместо тегов <TD> использованы теги <TH>, которые специально предназначены для выделения заголовков колонок таблицы (от *англ.* table header — заголовок таблицы). Одиночный тег
 заменяется на перевод строки и в приведенном примере способствует уменьшению ширины третьей колонки таблицы.

Некоторые теги допускают использование атрибутов, которые записываются после названия тега через пробел перед закрывающей угловой скобкой. Например, чтобы текст в ячейке таблицы был отцентрирован по горизонтали, можно использовать атрибут ALIGN со значением CENTER следующим образом:

```
<TD ALIGN="CENTER"> содержимое ячейки </TD>
```

Основные варианты стилевого оформления участков текста — выделение жирным шрифтом, курсивом и подчёркиванием — можно выполнить с помощью тегов <B> (от *англ*. bold — жирный), <I> (от *англ*. italic — курсивный) и <U> (от *англ*. underline — подчёркивание) следующим образом:

```
<P>В этом абзаце демонстрируется оформление <B>жирным шрифтом</B>,
<I>курсивом</I> и <U>подчёркиванием</U> участков текста.</P>
```

Отформатированный описанными способами текст на странице документации будет выглядеть, как показано на рисунке 7.3.

Глава первая

В кроличьей норе

Алисе надоело сидеть на пригорке рядом с сестрой и ничего не делать.

N п/п	Название предмета	Учебных часов
1	Математический анализ	86
2	Аналитическая геометрия	72
3	Программирование	80

В этом абзаце демонстрируется оформление **жирным шрифтом**, *курсивом* и подчёркиванием участков текста.

Рис. 7.3

Отображение отформатированного текста на Wiki-страничке

В документации к программному продукту наверняка встретятся участки кода, иллюстрирующие текстовое описание. Их надо оформлять с помощью тегов <NOWIKI>, <PRE> и <CODE> следующим образом:

```
<NOWIKI><PRE><CODE>
for i in range (8):
  for j in range (8):
    if (i + j) % 2:
      print (' .', end = '')
    else:
      print (' +', end = '')
  print ()
</CODE></PRE></NOWIKI>
```

В приведённом примере тег <NOWIKI> отключает внутри блока трактовку специальных комбинаций символов в качестве элементов разметки, в результате чего содержащийся в нём текст отображается как есть. Тег <PRE> сохраняет пробельные отступы и переносы строк, что особенно важно для языка программирования Python. А тег <CODE> информирует систему отображения, что в указанном блоке содержится текст на языке программирования. Обычно это приводит к тому, что текст отображается монотипным шрифтом.

Если в тексте, оформленном в соответствии с разметкой Wiki, надо явным образом представить знаки «меньше» или «больше», то, чтобы система не пере-

путала их с обозначением тега, их надо заменить специальными фразами: < (от *англ*. less than — меньше чем) и > (от *англ*. greater than — больше чем). Такая замена не требуется внутри блоков <NOWIKI>.

7.4. Язык разметки Markdown

Язык разметки Markdown был придуман в 2004 г. Джоном Грубером (John Gruber) и Аароном Шварцем (Aaron Swartz). Идея заключалась в таком обозначении структурных элементов текста (заголовков, абзацев, списков), чтобы эта разметка естественно выглядела в тексте при просмотре его в обычном текстовом редакторе, но при необходимости могла быть преобразована в оформление, соответствующее правилам HTML или других языков разметки.

Над синтаксисом языка Markdown его авторы работали совместно. В итоге Джон Грубер написал одноимённую утилиту, которая преобразовывала текст с разметкой Markdown в документ HTML, а Аарон Шварц — утилиту html2text, решающую обратную задачу.

По мнению авторов Markdown, для лучшего его понимания надо сразу рассмотреть пример текста, оформленного с его помощью. Последуем этому совету и оформим примеры из предыдущего раздела в стиле Markdown:

Листинг. Текст с разметкой в стиле Markdown

```
Глава первая
============

В кроличьей норе
----------------

  Алисе надоело сидеть на пригорке рядом с сестрой и ничего не делать.
Раз или два она заглянула украдкой в книгу, которую читала её сестра,
но там не было ни разговоров, ни картинок.

### Дама сдавала в багаж:

* картину;
* корзину;
* картонку;
* собачонку.

### Этапы жизненного цикла компьютерной программы:

1. Проектирование.
1. Реализация.
1. Внедрение.
1. Эксплуатация.


В этом абзаце демонстрируется оформление **жирным шрифтом** и
_курсивом_
участков текста.
```

~~~
| N п/п |  Название предмета    | Учебных часов |
-------------------------------------------------
|:1     :|Математический анализ  |:86           :|
|:2     :|Аналитическая геометрия|:72           :|
|:3     :|Программирование       |:80           :|

```````````````````````````
for i in range (8):
  for j in range (8):
    if (i + j) % 2:
      print (' .', end = '')
    else:
      print (' +', end = '')
  print ()
```````````````````````````
~~~

Как можно видеть, текст, оформленный в стиле Markdown, вполне читаем в своём первоначальном виде, и с первого взгляда трудно понять, где здесь вообще элементы разметки. Поэтому рассмотрим этот пример более подробно.

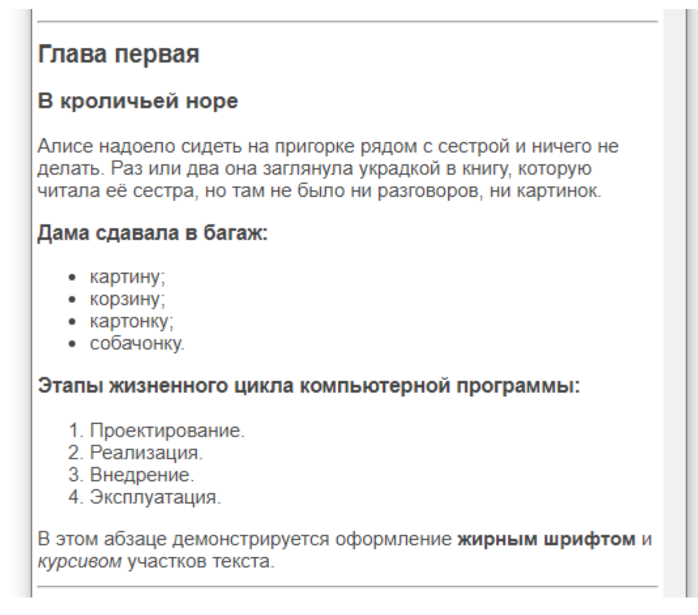
Начнём с заголовков. Заголовки первого уровня в разметке Markdown — это текст, подчёркнутый с помощью символов «равно», а заголовки второго уровня — текст, подчёркнутый с помощью символов «минус». В приведенном примере это «Глава первая» и «В кроличьей норе» соответственно. На самом деле, достаточно трёх подряд идущих символов подчёркивания, чтобы разметка сработала.

Заголовки следующих уровней обозначаются подряд идущими символами «решётка» перед текстом заголовка. Количество символов соответствует уровню заголовка. В приведенном примере есть два заголовка третьего уровня, это «Дама...» и «Этапы...». В принципе, заголовки первого и второго уровней можно тоже помечать соответственно одним и двумя символами «решётка», однако приведённый в примере вариант выглядит в тексте более наглядным и аккуратным. На веб-странице же оба варианта будут выглядеть одинаково.

Ненумерованный список, как и в Wiki-разметке, может быть создан путём размещения символов «звёздочка» перед его элементами. Допустимо также использовать символы «плюс» и «минус», если это улучшает вид исходного текста документа.

В начале строки любая цифра со следующей за ней точкой означает элемент нумерованного списка. Следить за нумерацией в исходном тексте нет необходимости — при обработке исходного текста будут автоматически проставлены возрастающие номера. Это продемонстрировано в приведенном примере, для чего перед каждым элементом нумерованного списка стоит обозначение «1.». На практике лучше и в исходном тексте использовать возрастающие номера, это сделает его более естественным. В нумерованных списках допускается замена точки на закрывающую круглую скобку.

Markdown позволяет выделить отдельные слова текста жирным шрифтом или курсивом. Для этого перед и после слова надо поставить признаки выделения: две подряд идущие звёздочки (жирный шрифт) или один символ подчёркивания (курсив). На рисунке 7.4 показано, как выглядит оформленный в стиле Markdown текст на веб-странице после применения стилей.



Глава первая

В кроличьей норе

Алисе надоело сидеть на пригорке рядом с сестрой и ничего не делать. Раз или два она заглянула украдкой в книгу, которую читала её сестра, но там не было ни разговоров, ни картинок.

Дама сдавала в багаж:

- картину;
- корзину;
- картонку;
- собачонку.

Этапы жизненного цикла компьютерной программы:

1. Проектирование.
2. Реализация.
3. Внедрение.
4. Эксплуатация.

В этом абзаце демонстрируется оформление **жирным шрифтом** и *курсивом* участков текста.

Рис. 7.4

Веб-страница, построенная на основе текста с разметкой Markdown

Таблицы тоже размечаются наглядным способом. Колонки разделяются символами вертикальной черты, заголовок таблицы отделяется от её тела горизонтальной линией из знаков «минус».

По умолчанию содержимое ячеек заголовка таблицы выравнивается по центру, а содержимое ячеек тела таблицы — по левому краю. Это можно изменить, если поставить двоеточие внутри ячейки рядом с той границей, к которой требуется выровнять содержимое. Если двоеточия стоят у обеих границ, то содержимое будет выровнено по центру.

Вставить листинг программы в текст с помощью Markdown очень просто. Достаточно начало и конец листинга обозначить линиями из обратных кавычек.

Отображение на веб-странице таблицы и листинга программы, размеченных с помощью Markdown, показано на рисунке 7.5.

N п/п	Название предмета	Учебных часов
1	Математический анализ	86
2	Аналитическая геометрия	72
3	Программирование	80

```
for i in range (8):
  for j in range (8):
    if (i + j) % 2:
      print (' .', end = '')
    else:
      print (' +', end = '')
  print ()
```

Рис. 7.5

Веб-страница с таблицей и листингом программы, размеченными с помощью Markdown

7.5. Доступ к системе документирования

Работа с документацией в системе Fossil осуществляется через веб-интерфейс. По умолчанию участники проекта могут только просматривать документацию. Чтобы увидеть список имеющихся страниц документации, надо воспользоваться пунктом горизонтального меню Wiki | List, пунктом раскрывающегося меню List of Wiki Pages или перейти по гиперссылке http://fossil.server:8081/wcontent (рис. 7.6).

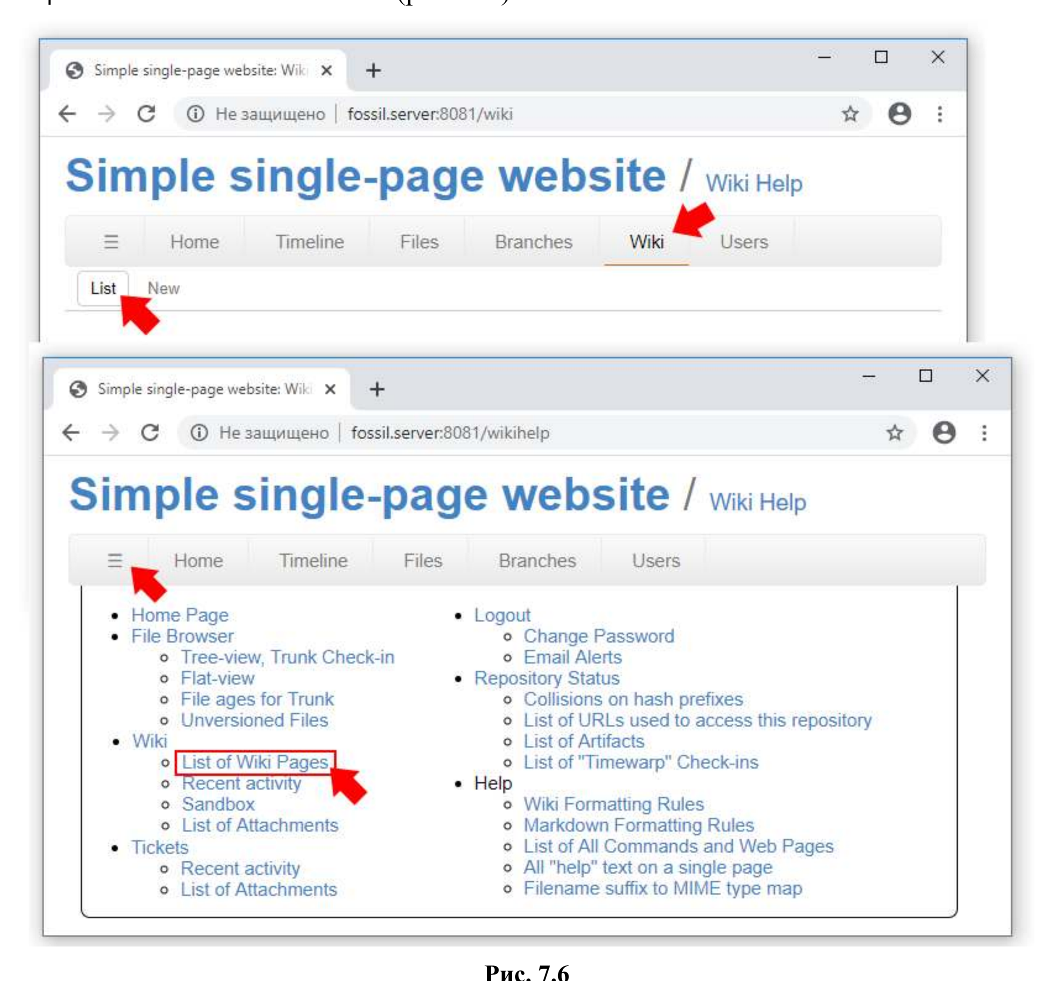


Рис. 7.6

Просмотр страниц документации

Чтобы создавать новые страницы документации и изменять содержимое существующих страниц, пользователь должен обладать необходимыми для этого полномочиями. Их может предоставить администратор системы путём отметки соответствующих пунктов на карточке учётной записи пользователя. К системе документирования относятся пять разрешений (рис. 7.7):

- Read Wiki — чтение документации;
- New Wiki — создание новой страницы документации;

- Append Wiki — дополнение существующей страницы документации;
- Write Wiki — написание документации;
- Moderate Wiki — модерирование документации.

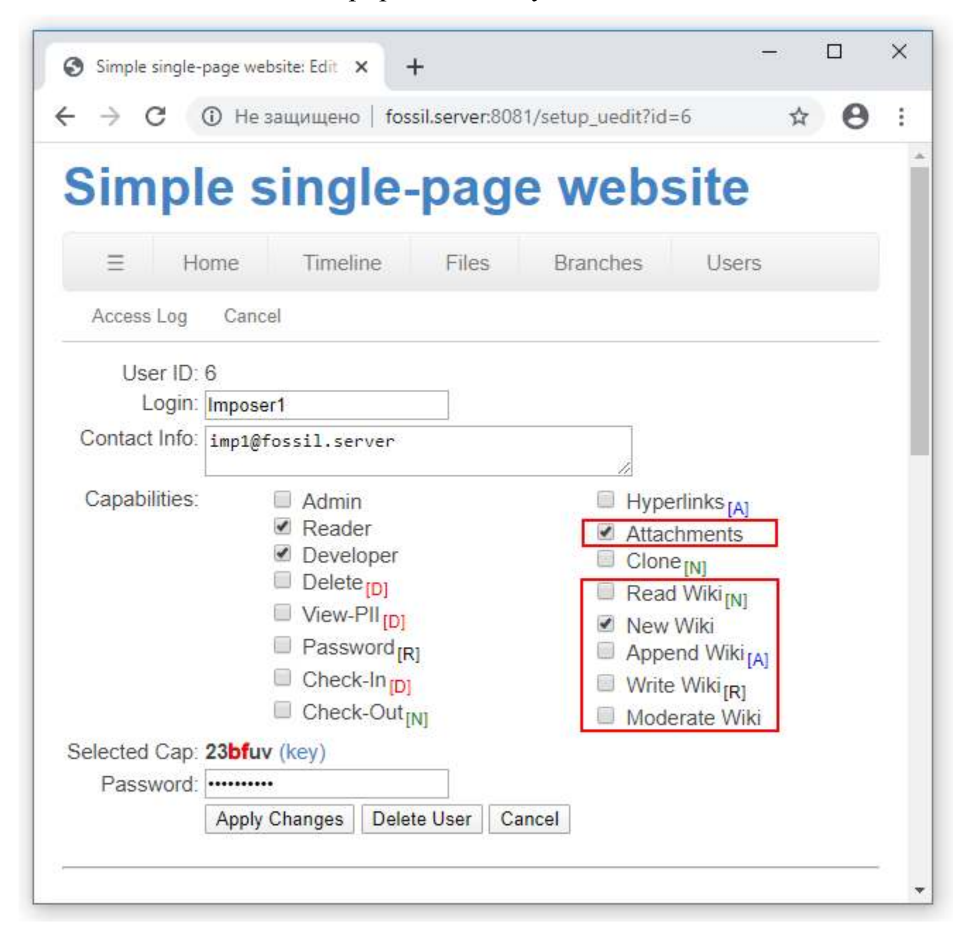


Рис. 7.7

Права доступа к документации. Чтобы разработчики проекта могли создавать новые страницы документации и прикреплять к ним файлы, им надо предоставить разрешения New Wiki и Attachments

Разрешение на чтение документации (Read Wiki, код j) имеется у группы [N]obody, к которой относятся все посетители веб-интерфейса Fossil, поэтому отдельно его предоставлять необходимости нет.

Пользователи, входящие в группу [R]eader, автоматически приобретают право на редактирование существующих страниц документации (Write Wiki, код k).

Анонимные пользователи, входящие в группу [A]nonymous, к которым относятся любые пользователи, авторизовавшиеся в системе, тоже имеют возможность оставить свой след в документировании проекта. У них есть разре-

шение на дополнение существующей страницы документации (Append Wiki, код m), позволяющее им дописывать свой текст внизу страницы.

Создать новую страницу документации рядовой участник проекта, которому обычно назначаются группы разрешений [R]eader и [D]eveloper, не сможет, если ему не будет явно предоставлено разрешение New Wiki (код f). А чтобы иметь возможность дополнять страницы документации файлами, например с двоичными данными или иллюстрациями, потребуется ещё разрешение Attachments (код b).

Если в параметры системы документирования Fossil специально не вносились изменения, то в полномочии на модерирование страниц документации (Moderate Wiki) нет необходимости. Новые страницы и внесённые в них изменения становятся общедоступными автоматически, без предварительной проверки авторизованным пользователем.

7.6. Создание и редактирование статей

Сначала список страниц документации пуст. Чтобы создать новую статью, надо воспользоваться пунктом дополнительного горизонтального меню New (Новая) на странице веб-интерфейса Wiki либо перейти по гиперссылке Create a new wiki page (Создать новую страницу документации) там же (рис. 7.8).

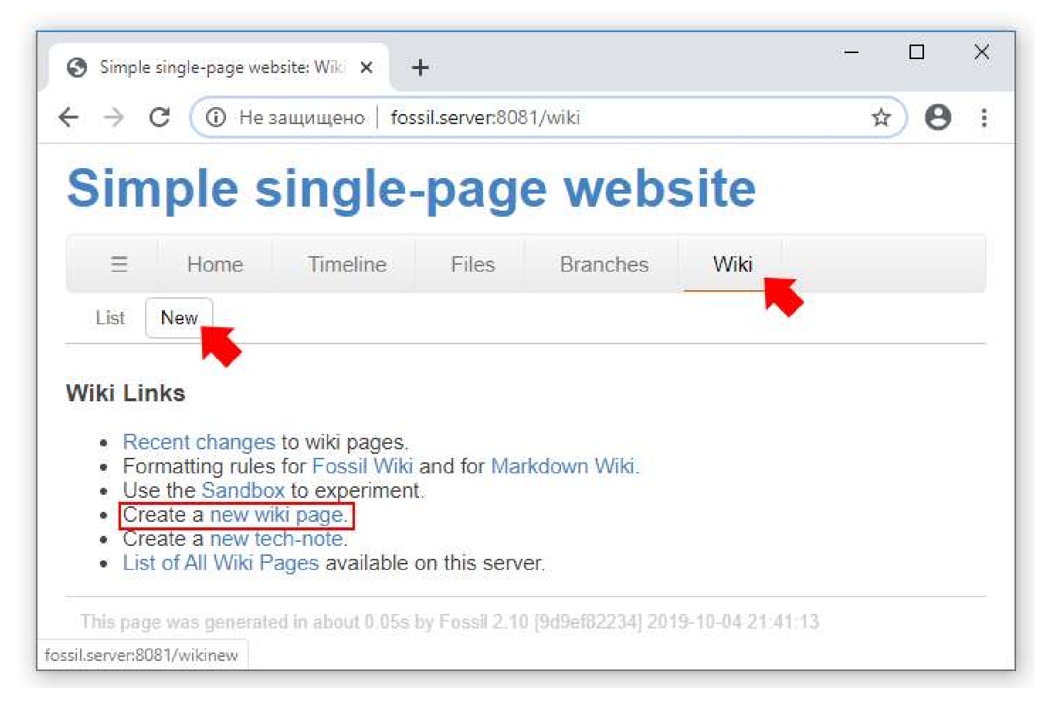


Рис. 7.8

Переход к созданию новой страницы документации

После этого будет предложено заполнить простую форму из двух полей со сведениями о создаваемой веб-странице (рис. 7.9). В первом поле надо ввести

название страницы, которое в дальнейшем будет отображаться в списке страниц документации. На название накладываются следующие ограничения:

- название должно содержать как минимум один, но не более ста символов;
- оно не может начинаться или заканчиваться пробелами;
- название не должно содержать управляющие символы (табуляцию, перевод строки и др.);
- между словами в названии не должно быть более чем одного пробела.

Если будет введён текст, в точности повторяющий название уже существующей страницы, то новая страница создана не будет, а будет предложена к редактированию одноимённая существующая страница.

Во втором поле предлагается выбрать из списка язык разметки, который будет использоваться для оформления создаваемой страницы. Для выбора доступны следующие варианты:

- Fossil Wiki;
- Markdown;
- Plain Text.

Первые два языка разметки были подробно рассмотрены выше, а третий означает обычный текст без форматирования, т. е. к набранному тексту не будет применяться никакое стилевое оформление.

Для перехода к набору содержимого страницы документации надо нажать кнопку Create, находящуюся внизу формы.

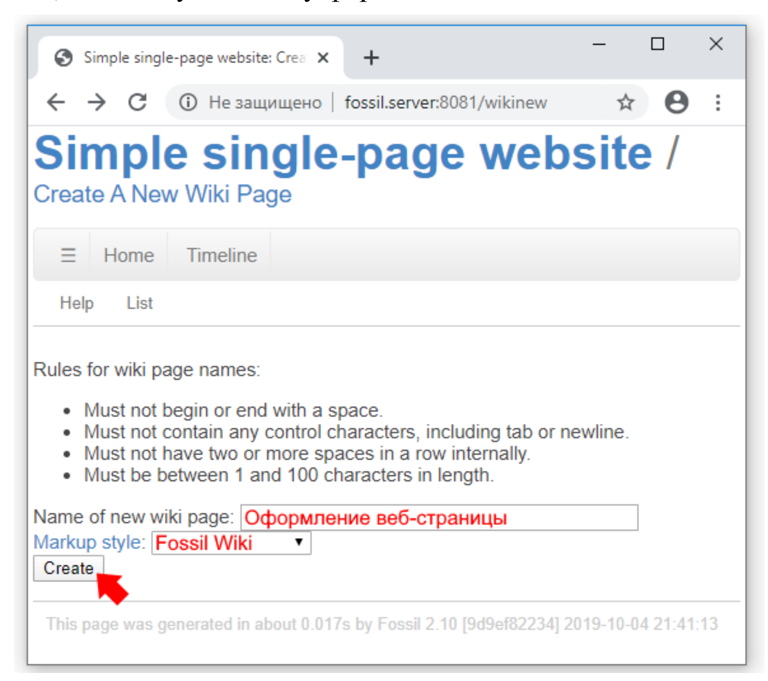


Рис. 7.9

Заполнение сведений о странице документации

Открывшаяся форма (рис. 7.10) позволяет изменить язык разметки, используемый для оформления страницы документации, путём выбора нового значения из списка в поле Markup style (Стиль разметки).

Содержимое страницы документации набирается в большом текстовом поле с использованием выбранного языка разметки. В приведённом примере текст оформлен с помощью Markdown. Когда текст будет набран, надо нажать кнопку Preview Your Changes (Предпросмотр ваших изменений), находящуюся внизу формы. Над формой отобразится текст с применением форматирования, заданного языком разметки, — таким же его будут в дальнейшем видеть пользователи. А рядом с нажатой кнопкой появится другая — Apply These Changes (Применить эти изменения). Если внешний вид документа устраивает, её можно нажать для выхода в режим просмотра страницы из режима её правки с сохранением выполненных изменений. Кнопка Cancel позволяет выйти из режима правки без сохранения изменений.

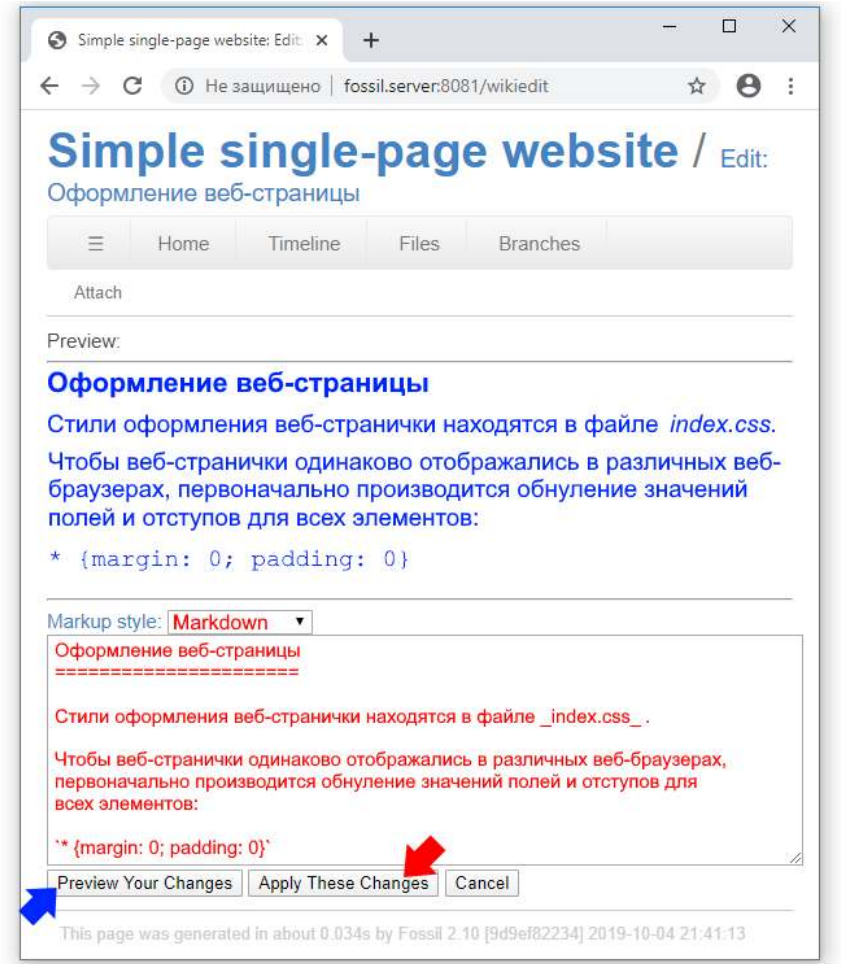


Рис. 7.10

Заполнение содержимого страницы документации

Чтобы исправить имеющуюся страницу документации, надо открыть её для просмотра, перейдя по гиперссылке с её названием в списке страниц (рис. 7.11). После этого в дополнительном горизонтальном меню, находящемся над содержимым страницы, надо выбрать пункт Edit.

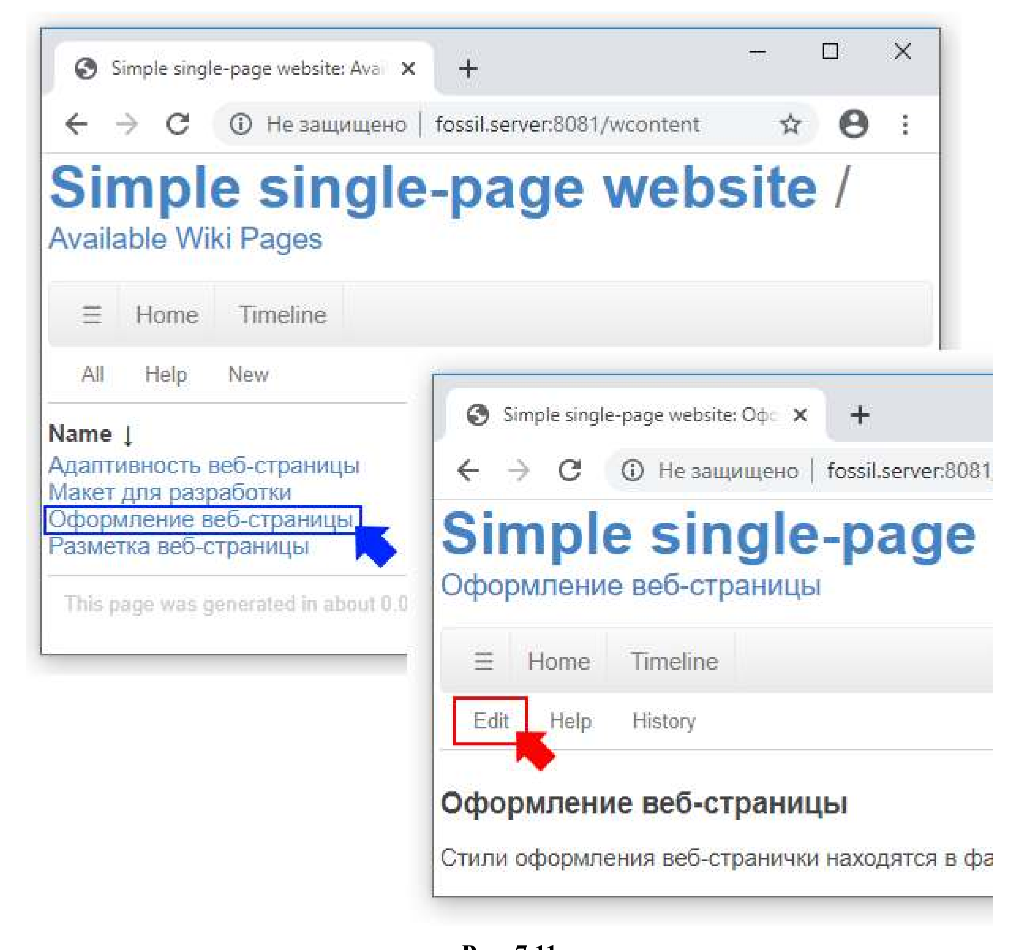


Рис. 7.11

Исправление страницы документации

Интерфейс и процедура редактирования страницы документации полностью аналогичны таковым при недавно описанном создании новой страницы.

7.7. Добавление файлов и изображений

Если у пользователя имеется разрешение на прикрепление файлов (Attachments), то над формой в дополнительном горизонтальном меню будет присутствовать пункт Attach. С его помощью можно, например, прикрепить файл с иллюстрацией (рис. 7.12).

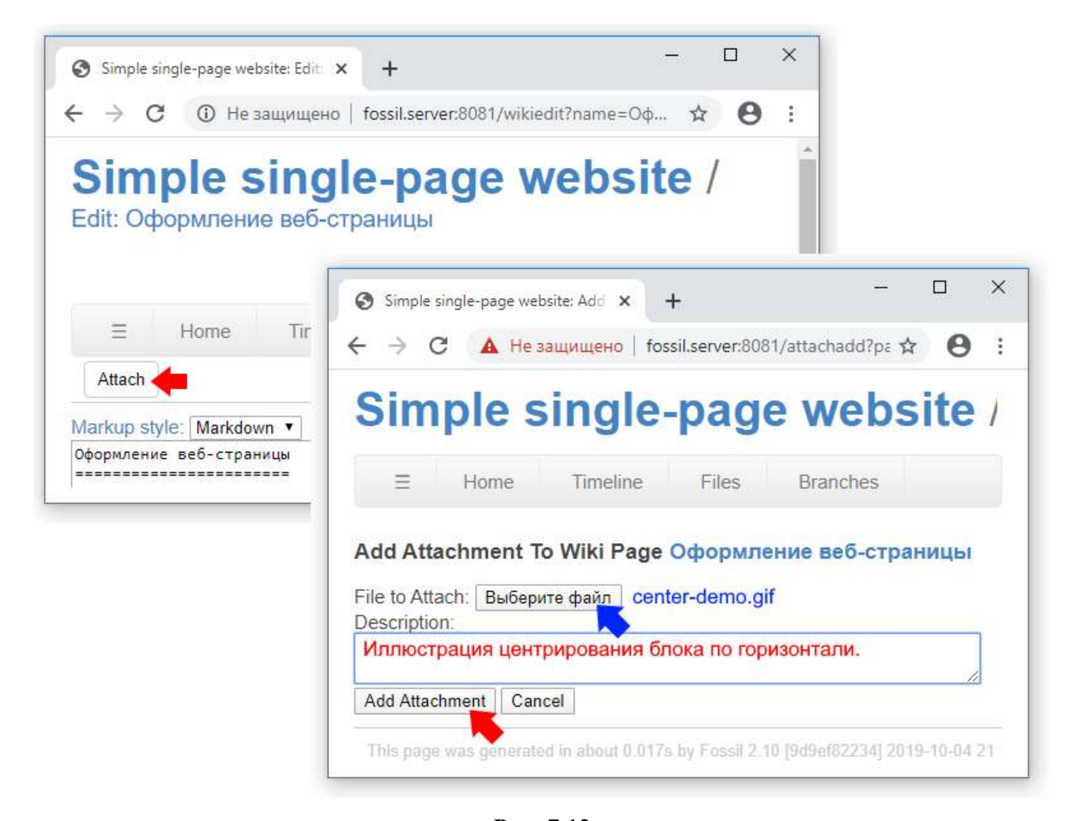


Рис. 7.12

Прикрепление файла к странице документации

Для этого на открывшейся форме надо нажать кнопку Выберите файл и указать на добавляемый файл в стандартном окне диалога выбора файла. Добавляемый файл можно снабдить описанием, заполнив текстовое поле Description. После этого для прикрепления выбранного файла надо нажать кнопку Add Attachment (Добавить вложение). Отказаться от выполнения операции можно с помощью кнопки Cancel (Отменить).

Добавленные файлы попадают в блок Attachments, находящийся в нижней части страницы документации. По гиперссылке details открывается страничка с описанием вложения.

Добавленное изображение можно использовать на странице документации. Для этого в языке разметки Fossil Wiki надо воспользоваться тегом <IMG SRC="Адрес изображения" ALT="Описание изображения">, а в Markdown имеется такая конструкция:

```
![Описание изображения](Адрес изображения)
```

После восклицательного знака в квадратных скобках указывается текстовое описание вставляемого изображения, а следом за ним в круглых скобках — его адрес.

Если изображение находится на внешнем сервере в Интернете, в качестве адреса надо указать его URL. А для ссылки на файл, хранящийся в системе

Fossil, достаточно указать его идентификатор. Узнать идентификатор прикреплённого изображения можно на посвящённой ему страничке. Чтобы на неё попасть, надо в режиме просмотра страницы документации перейти по гиперссылке, представленной именем прикреплённого файла, как показано на рисунке 7.13.

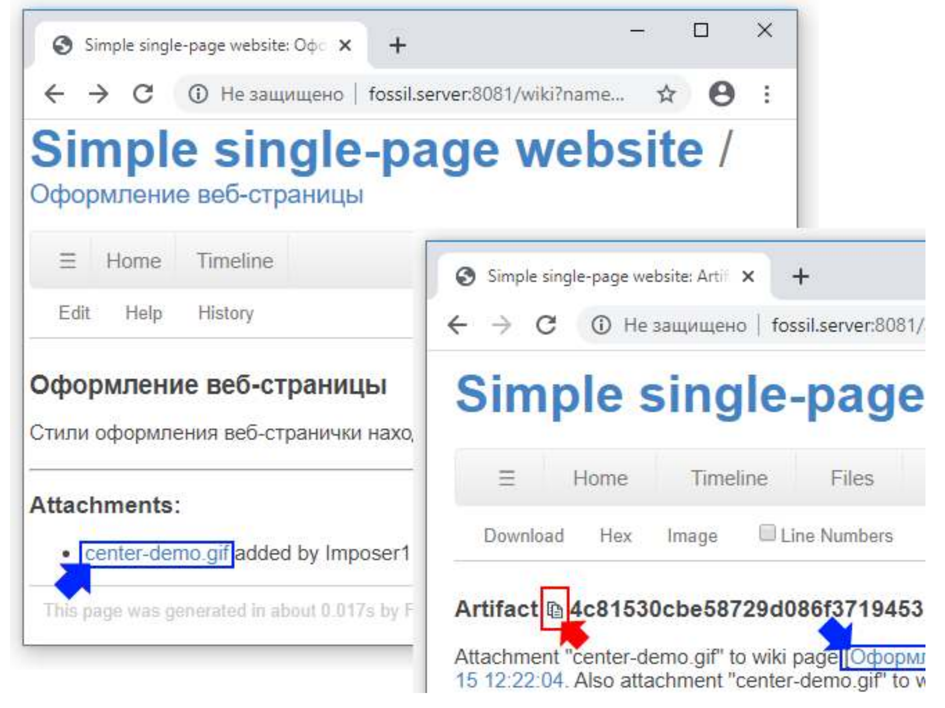


Рис. 7.13

Для получения дополнительной информации о прикреплённом файле надо на странице документации перейти по гиперссылке, представленной его именем. Скопировать идентификатор файла можно с помощью пиктограммы Копировать

В верхней части открывшейся страницы, посвящённой прикреплённому файлу, имеется дополнительное горизонтальное меню, содержащее пункты:

- Download — скачать файл на локальный компьютер;
- Hex — просмотреть содержимое файла в шестнадцатеричном виде;
- Image — просмотреть файл как изображение;
- Line Numbers — если файл текстовый, то включить для него отображение номеров строк.

Под этим меню размещена строка с полным идентификатором файла в системе Fossil. Чтобы скопировать в буфер обмена первые символы идентификатора, достаточные для его указания в ссылке на файл, можно воспользоваться пиктограммой Копировать. Для быстрого возврата отсюда на страницу документации служит гиперссылка с её названием (в приведенном примере — Оформление веб-страницы).

Ссылка на прикреплённый файл должна быть записана в виде: /raw/идентификатор. Для приведенного примера конструкция вставки иллюстрации в документ примет вид:

```
![Иллюстрация центрирования блока по горизонтали](/raw/4c81530cbe58729d)
```

Результат такого оформления показан на рисунке 7.14.

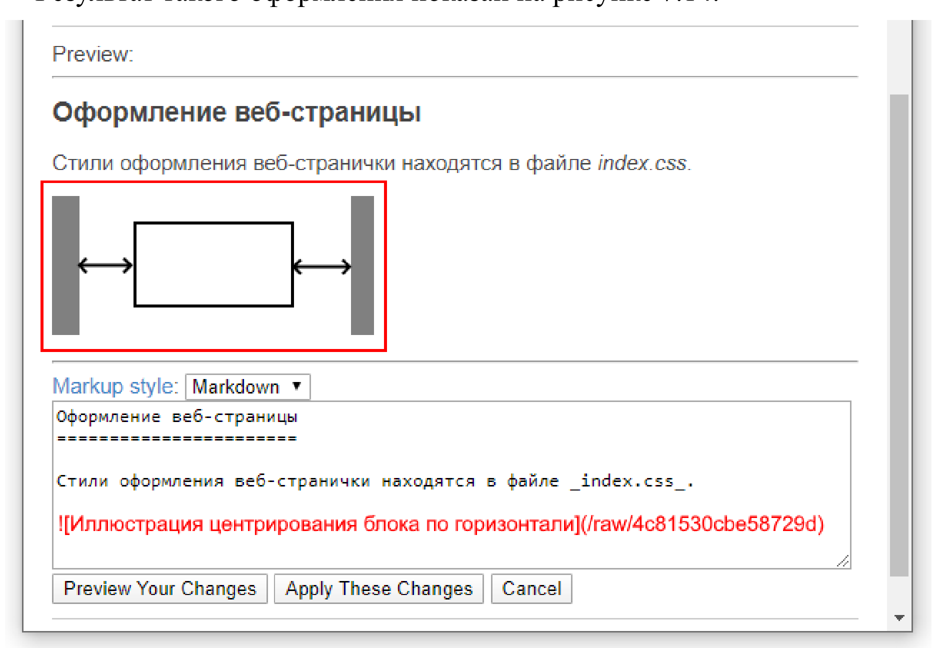


Рис. 7.14

Страница документации с иллюстрацией

Ошибочно прикреплённый файл можно удалить. Для этого надо перейти по гиперссылке details (подробности), имеющейся на странице документации под её основным содержимым в информационном блоке со списком прикреплённых файлов. Открывшаяся страница, помимо детальной информации о файле, содержит горизонтальное меню, в котором есть пункт Delete (Удалить). При выборе этого пункта будет выведен запрос подтверждения Confirm you want to delete the attachment shown below (Подтвердите ваше желание удалить вложение, показанное ниже), на который можно утвердительно ответить нажатием имеющейся рядом с ним кнопки Confirm (Подтвердить). После этого файл будет удалён.

7.8. Технические заметки

Рассмотренные выше страницы документации позволяют создать подборку статей, посвящённых различным аспектам реализуемого проекта. Они собираются в каталог, доступный через пункт меню Wiki | List, в котором представлены их названия. Такая организация сведений удобна для хранения информации в разрезе тем и напоминает алфавитный блокнот.

Однако в некоторых случаях информацию удобнее хранить в хронологическом порядке. На этом принципе построены блокноты другого типа — ежедневники. Технологичным аналогом ежедневников являются «живые журналы» — блоги.

Для ведения дневника проекта в системе Fossil предлагается использовать специальный тип страниц документации — технические заметки (technical notes). Эти страницы идентифицируются не своими заголовками, как рассмотренные ранее, а отметками времени, как точки сохранения файлов проекта. Поэтому логично, что для их просмотра надо перейти на шкалу времени проекта (с помощью пункта Timeline основного горизонтального меню) и выбрать категорию Tech Notes (по умолчанию на шкале времени отображаются только точки сохранения — Check-ins), как показано на рисунке 7.15.

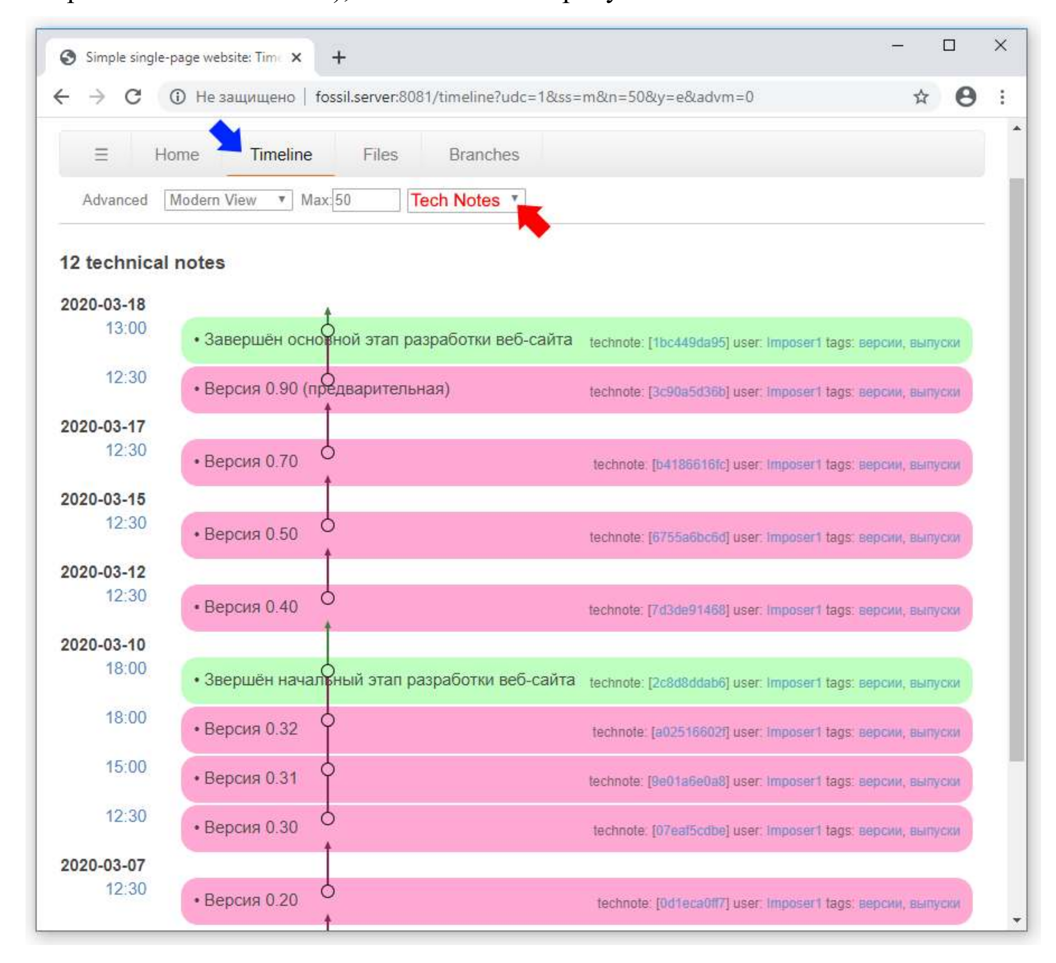


Рис. 7.15

Просмотр технических заметок на шкале времени

Для таких заметок сразу можно предложить несколько полезных применений:

- перечисление задач, которые требуется решить по плану реализации проекта (при создании заметки можно указывать любую дату, в том числе будущую);
- отмечание этапов реализации проекта, истории смены версий реализуемого продукта;
- фиксирование моментов смены инструментария, используемого в ходе разработки (версии компилятора, системы сборки, программных библиотек);
- размещение новостных сообщений, консолидирующих информацию о текущем состоянии проекта.

Поскольку технические заметки в Fossil — это просто иначе организованная категория страниц документации, для работы с ними (просмотра, создания и редактирования) пользователям системы необходимы те же полномочия, что уже были рассмотрены выше. А для оформления технических заметок используются рассмотренные ранее языки разметки Fossil Wiki и Markdown.

Для создания новой заметки надо воспользоваться пунктом главного меню Wiki, после чего перейти по гиперссылке Create a new tech-note., или сразу перейти по адресу http://fossil.server:8081/technoteedit (рис. 7.16).

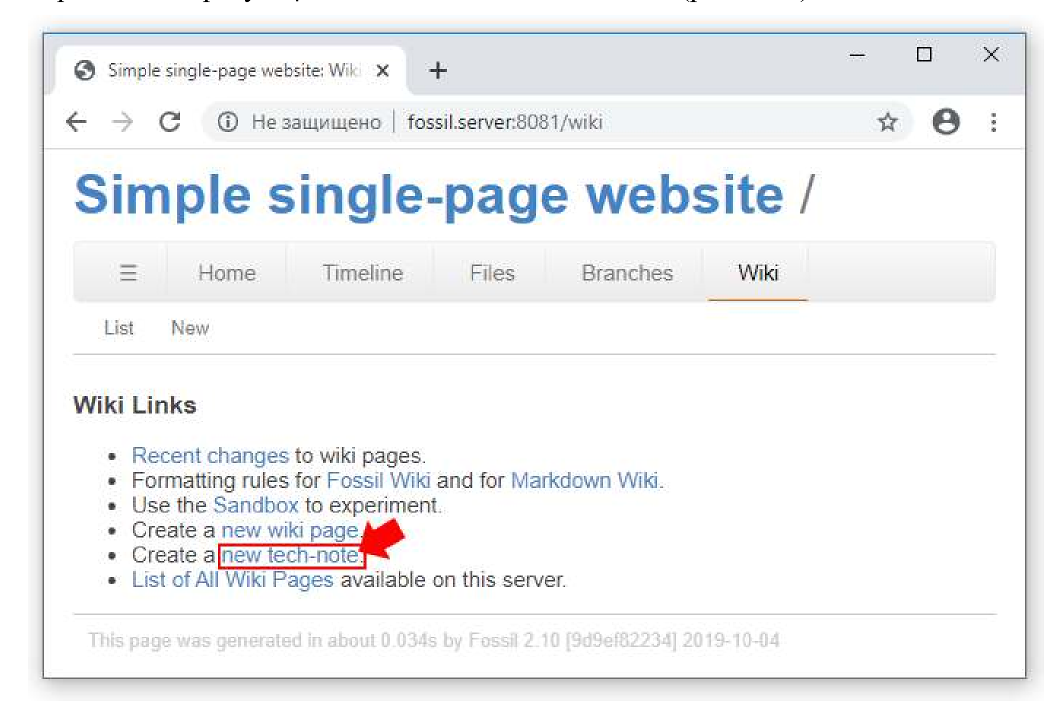


Рис. 7.16

Переход к созданию технической заметки

Откроется форма для создания заметки, на которой предлагается заполнить следующие поля (рис. 7.17):

- Timestamp (UTC) — отметка времени заметки, в соответствии с которой она будет размещена на шкале времени проекта;

- Timeline Comment — описание заметки, аналог заголовка страницы документации;
- Timeline Background Color — при установленном флажке Use custom color описание заметки будет отображаться на шкале времени с указанным фоновым цветом, который можно выбрать из палитры (удобно использовать для визуального разделения заметок по категориям);
- Tags — ярлыки для классификации заметок;
- Markup Style — язык разметки, который будет использован для оформления содержимого заметки;
- Page Content — текст, составляющий подробное содержимое технической заметки.

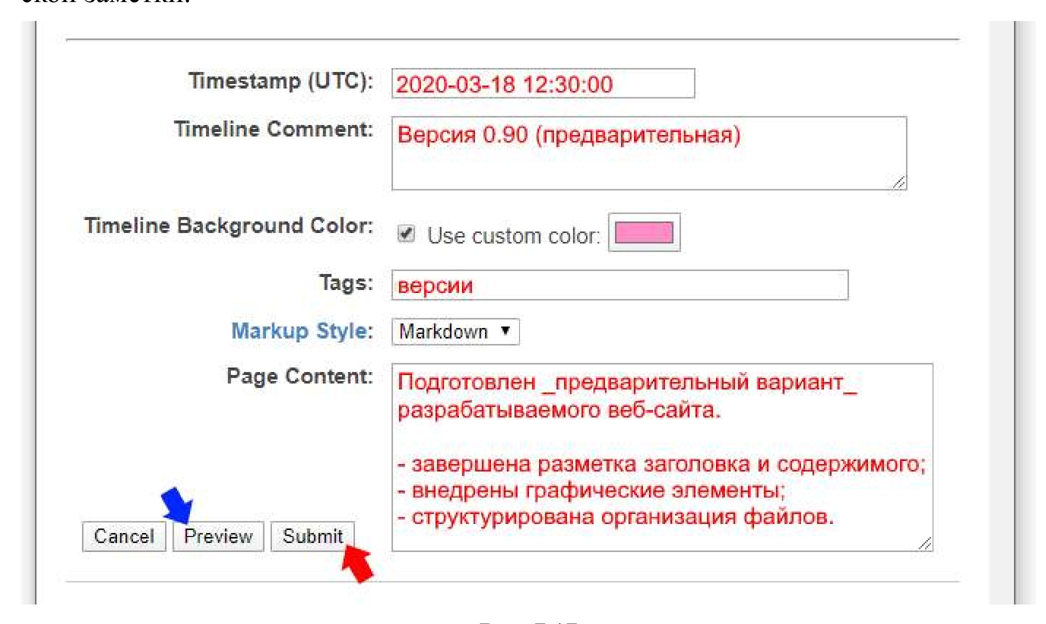


Рис. 7.17

Создание технической заметки

После заполнения представленных на форме полей надо нажать кнопку Preview (Предпросмотр), находящуюся внизу формы. Это приведёт к отображению над формой описания и содержимого создаваемой технической заметки для предварительного просмотра (рис. 7.18). А рядом с кнопкой Preview появится кнопка Submit (Подтвердить), которую надо нажать для завершения процедуры создания заметки, если просмотренный результат устраивает.

На странице с текстом содержимого заметки, к которой можно перейти со шкалы времени по гиперссылке с её идентификатором, имеется дополнительное горизонатльное меню, позволяющее выполнять над ней следующие действия (рис. 7.19):

- Attach — прикрепить к заметке файл (например, иллюстрацию);
- Context — переключиться на шкалу времени к отметке, указанной в заметке;

- Detail — просмотреть подробные сведения о заметке;
- Edit — внести изменения в заметку.

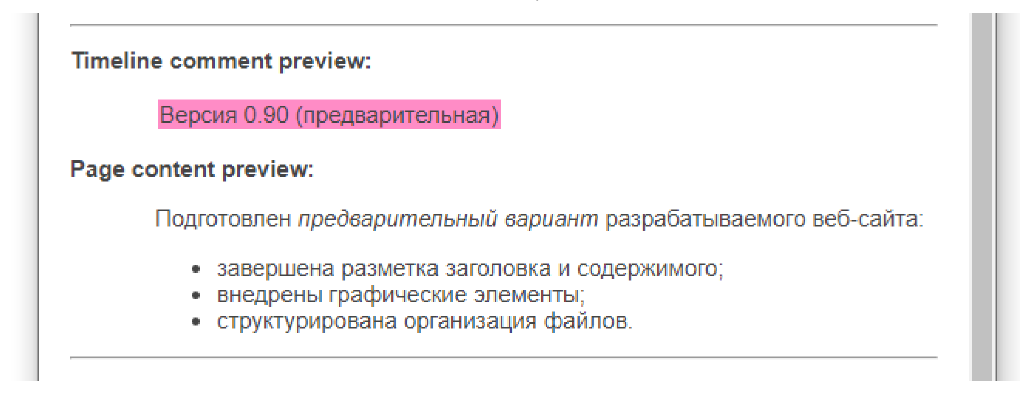


Рис. 7.18

Предварительный просмотр создаваемой заметки

Рис. 7.19

Меню технической заметки

При редактировании заметки надо особое внимание уделять флажку Timeline Background Color, если при создании указывался цвет фона. На форме редактирования он всегда снят, и если забыть его установить, то исправленная заметка будет отображаться без выделения цветом.

7.9. Дополнительные возможности

Рассмотренная система документирования уже сама по себе является полезным инструментом. Однако то, что она входит в состав Fossil в качестве одной из подсистем, открывает дополнительные возможности её применения.

Если при создании страницы документации указать в качестве её названия строку, составленную по шаблону: branch/НазваниеВетвиРепозитория, то содержимое этой страницы будет отображаться в заголовке шкалы времени при просмотре ветви с указанным названием. На рисунке 7.20 показано, как эта возможность использована для размещения комментария к ветви «заголовок».

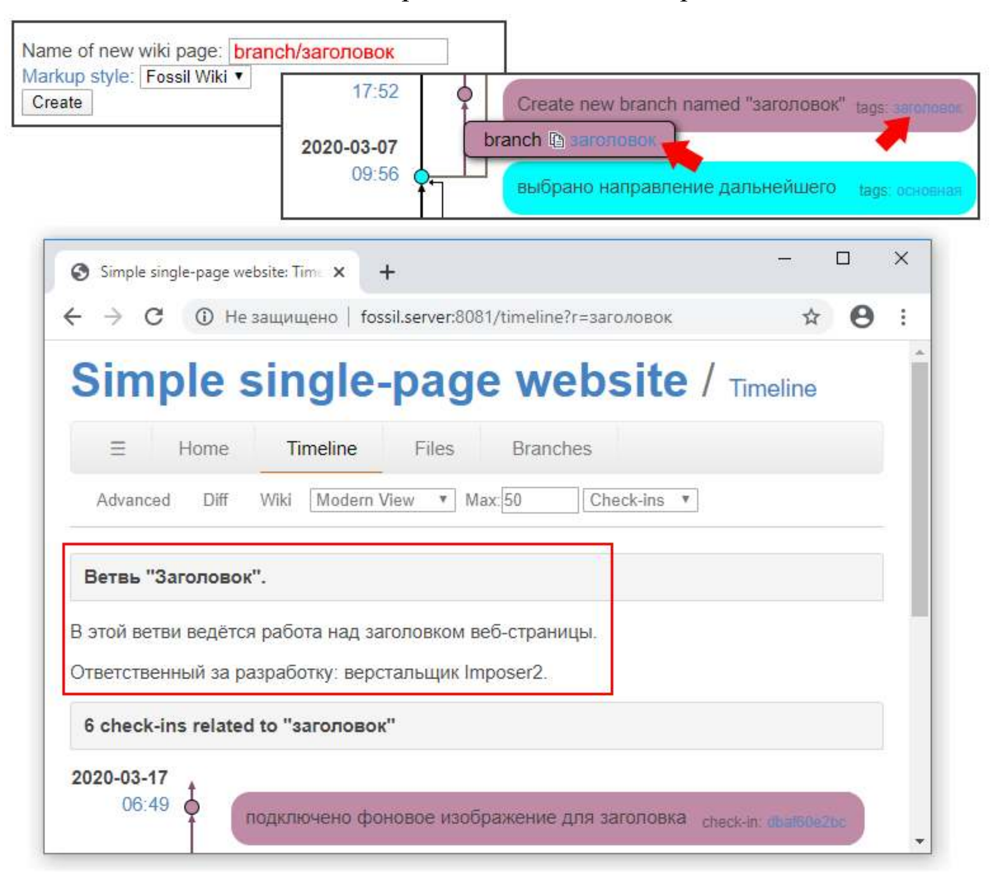
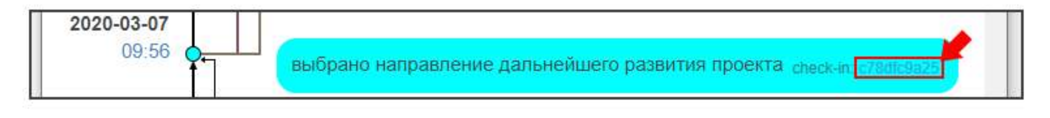


Рис. 7.20

Использование страницы документации в качестве комментария к ветви репозитория

Такие комментарии могут быть особенно полезны для аннотирования ветвей репозитория, созданных для проверки тех или иных идей. В них можно описать предпосылки для проведения испытания и сообщить полученный в итоге результат. Впоследствии эта информация может представлять историческую ценность для сотрудников, давно работающих над проектом или только входящих в курс дела.

Подобным образом можно добавить развёрнутое описание к конкретной точке сохранения в репозитории проекта. Для этого название страницы документации надо составлять по такому шаблону: checkin/ИдентификаторТочкиСохранения. Чтобы магия сработала, идентификатор точки сохранения должен быть указан полностью, все 64 шестнадцатеричные цифры. Альтернативный вариант — воспользоваться гиперссылкой Add Wiki:this checkin на странице с детальным описанием точки сохранения (рис. 7.21). Кстати, рядом с ней имеется гиперссылка с названием ветви репозитория, позволяющая сразу перейти к созданию описания для всей ветви, о чём было сказано выше.



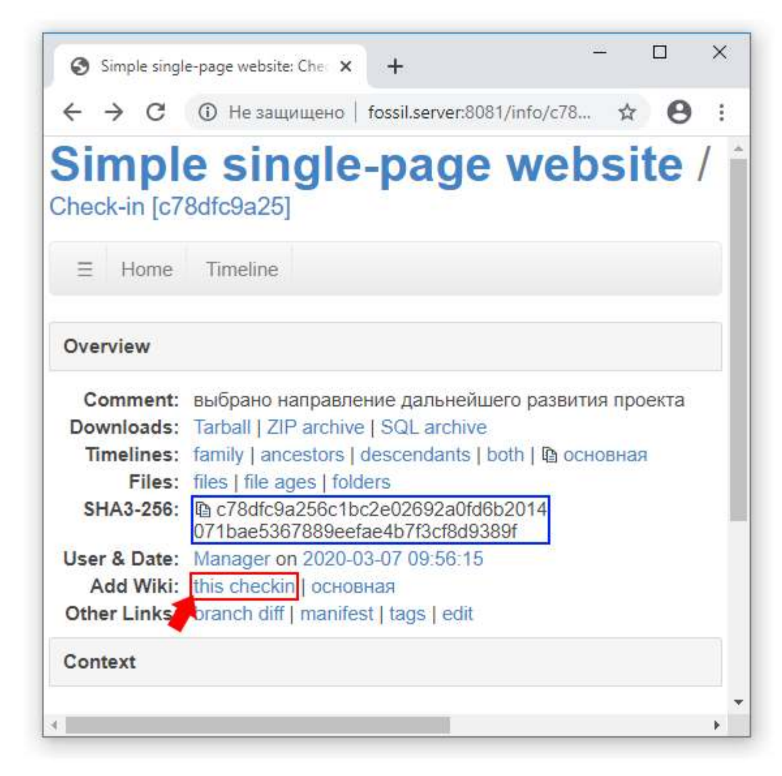


Рис. 7.21

Создание развёрнутого комментария к точке сохранения

Содержимое созданной по таким правилам страницы документации будет отображаться в заголовке страницы с детальной информацией о точке сохранения и является удобным средством для дополнения краткого комментария, которым обычно сопровождаются точки сохранения при выполнении операции fossil commit.

Несмотря на наличие полезных особенностей у страниц документации, созданных по описанным выше правилам, они всё равно остаются обычными

wiki-страницами, которые отображаются в списке наряду с остальными и точно так же могут быть просмотрены, отредактированы и снабжены иллюстрациями.

Страницами документации можно управлять не только через веб-интерфейс, но и посредством командной строки. Если подготовить текст с использованием языка разметки Markdown и сохранить его в каталоге ..\wiki в файле, например, с именем style.md, то создать в репозитории Fossil страницу документации на его основе можно с помощью команды (предполагается, что текущим является рабочий каталог проекта):

```
fossil wiki create "Стилевое оформление" ..\wiki\style.md -M markdown
```

В этой команде «Стилевое оформление» — это название страницы документации, а с помощью параметра *-M* указан использованный язык разметки — Markdown (допустимы ещё варианты fossil — для разметки Fossil-Wiki — и plain — для простого неформатированного текста). В случае успешного выполнения команды на экране появится сообщение:

```
Created new wiki page Стилевое оформление. (Создана новая страница
документации Стилевое оформление).
```

Оно означает, что страница документации создана в локальном репозитории. В сетевой репозиторий она попадёт в процессе синхронизации, которую можно инициировать с помощью команды fossil push или fossil sync. Если страницы документации создаются или корректируются через веб-интерфейс непосредственно в сетевом репозитории, то в локальные репозитории они попадут тоже после синхронизации, инициированной командой fossil pull или fossil sync. Посмотреть перечень страниц документации, имеющихся в локальном репозитории, можно с помощью команды:

```
fossil wiki list
```

Возможность управления страницами документации через интерфейс командной строки может пригодиться при автоматизации создания технических заметок с помощью сценариев командной оболочки. Для создания технической заметки используется команда:

```
fossil wiki create "Версия 1.00 (выпуск 1)" ..\notes\version.wiki
-t 2020-04-21T12:30:00 -M fossil
--technote-tags "версии,выпуски"
--technote-bgcolor cyan
```

В приведенном примере создаваемая техническая заметка будет датирована 21 апреля 2020 г., озаглавлена «Версия 1.00 (выпуск 1)», снабжена тегами «версии» и «выпуски» и выделена цветом фона cyan на шкале времени. Об успешном завершении команды свидетельствует сообщение:

```
Created new tech note 2020-04-21 12:30:00. (Создана новая техниче-
ская заметка 2020-04-21 12:30:00.)
```

Если страница документации или техническая заметка уже существует, то внести в неё изменение можно, заменив в приведенных примерах команду create на commit.

Создание и изменение страниц документации из командной строки могут осуществлять все пользователи Fossil, имеющие право на запись в репозиторий сервера (отметка «v» Developer или «i» Check-In на странице настроек пользователя). Специальных разрешений на работу со страницами документации, как для использования веб-интерфейса, здесь не требуется.

Помимо подсистемы Wiki в Fossil реализована возможность ведения документации в файлах рабочего каталога проекта. Поскольку эти файлы хранятся непосредственно в репозитории вместе с остальными файлами разрабатываемого проекта, такая документация называется встроенной (builtin). Этот способ документирования рассматривается в главе, посвящённой серверу Fossil, где с его помощью совершенствуется веб-интерфейс.

Глава 8
ФОРУМ

8.1. Виды и средства коммуникации

В коллективной работе важную роль играют коммуникации между участниками проекта. В процессе общения происходят синхронизация осведомлённости о состоянии проекта, обмен опытом и знаниями, возникает чувство локтя, снижается психологическая напряжённость. Коммуникации можно разделить на два основных вида: прямые и косвенные.

Прямая коммуникация — это непосредственное очное общение между людьми лицом к лицу. Это наиболее полноценная форма общения, при которой между участниками происходит не только обмен сухой информацией в вербальной форме, но и передача сигналов с помощью мимики, жестов, интонаций. Прямая коммуникация возможна лишь на небольших расстояниях и происходит только в настоящем времени. Наряду с положительными сторонами надо отметить, что оживлённое обсуждение сотрудниками своей проблемы может отвлекать находящихся рядом коллег от работы, мешать сосредоточиться.

Косвенная коммуникация — это любые формы общения между людьми, когда они находятся вне прямой видимости и слышимости. Очевидно, что при этом возникает потребность в использовании промежуточных носителей информации и вспомогательных средств для её выражения. Раньше наиболее характерной формой косвенной коммуникации были письма (с использованием бумаги, пера и чернил), телеграммы и телефонные разговоры (с применением электрических сигналов и аппаратов для кодирования сообщений). С появлением компьютерных сетей спектр средств косвенной коммуникации значительно изменился и расширился. На сегодняшний день важное место в жизни людей занимает общение в социальных сетях. Только косвенная коммуникация позволяет передавать информацию во времени из прошлого в будущее.

По числу участников средства косвенной коммуникации можно разделить на публичные (когда обсуждение доступно для широкого круга участников) и частные (когда круг участников намеренно ограничен). По оперативности средства коммуникации можно разделить на мгновенные (требующие постоянного внимания и незамедлительной реакции участников) и отложенные (когда участники обсуждения могут знакомиться с ним и выражать своё мнение в удобное для них время).

8.2. Доступ к обсуждениям

В системе Fossil в качестве встроенного средства коммуникации между участниками проекта предлагается использовать форум — публичное обсуждение тем, интересных участникам, в удобное для них время. Форум не может заменить другие средства общения, такие как электронная почта или чат, но спо-

собен хорошо их дополнить, выступая в качестве канала для неформальных служебных записок.

Участник проекта должен запросить права на доступ к форуму у администратора системы. Администратор может установить на карточке пользователя следующие отметки (рис. 8.1):

- Read Forum (код 2) — разрешение, предоставляющее право просмотра обсуждений;
- Write Forum (код 3) — разрешение, предоставляющее право создания тем и написания ответов;
- Write Trusted Forum (код 4) — знак особого доверия к пользователю, благодаря которому его сообщения не требуют проверки и одобрения модератором;
- Moderate Forum (код 5) — полномочия модератора, позволяющие просматривать и принимать решения (одобрить или отклонить) в отношении новых сообщений, написанных другими пользователями;
- Supervise Forum (код 6) — право назначения участникам форума отметки Write Trusted Forum, рассмотренной выше.

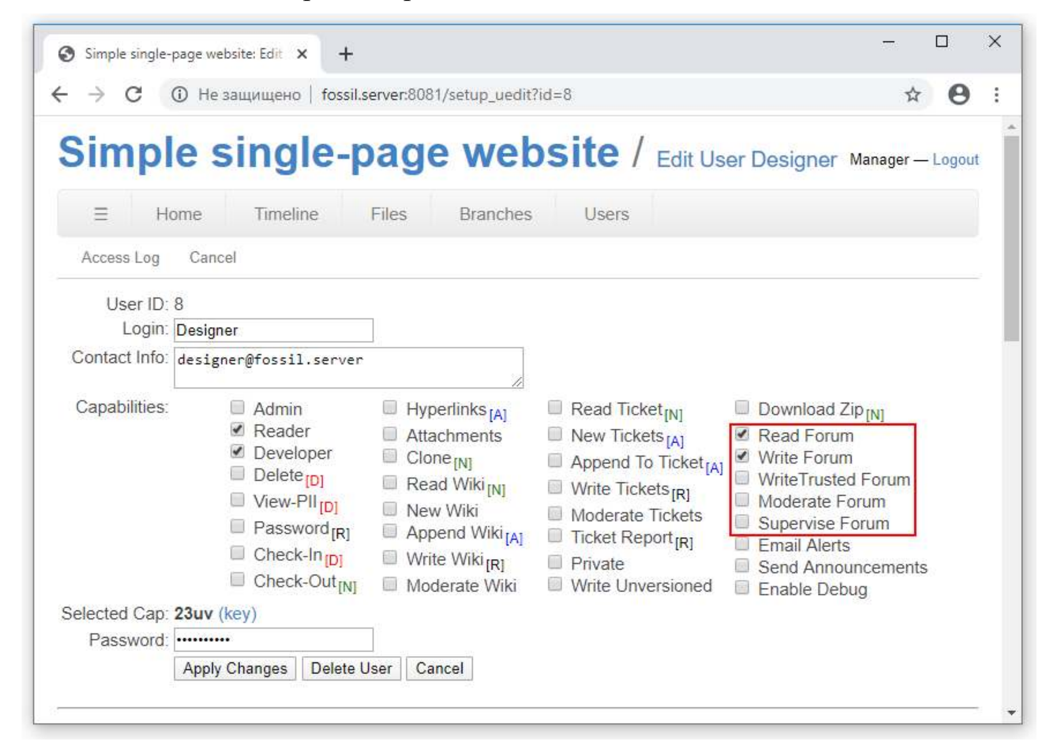


Рис. 8.1

Разрешения для доступа к форуму

Администраторы системы Fossil получают все перечисленные разрешения автоматически, поэтому пользователь Manager сразу имеет возможность выступать в роли модератора, что и будет использовано в рассматриваемом примере.

Чтобы попасть на форум, надо авторизоваться в системе Fossil, используя свои имя и пароль пользователя, и выбрать пункт главного горизонтального меню Forum (рис. 8.2).

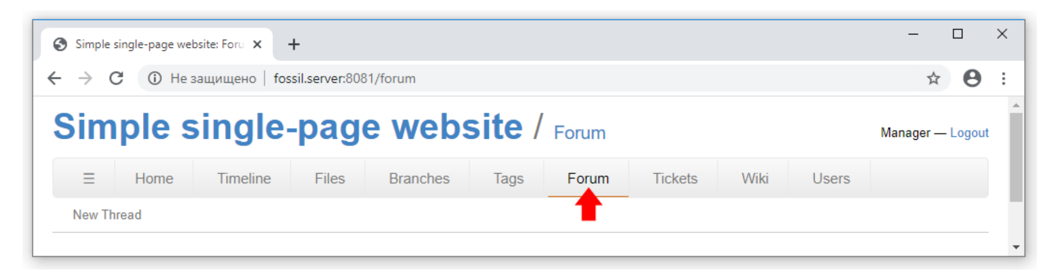


Рис. 8.2

Переход на веб-страницу форума

В некоторых случаях веб-интерфейс Fossil отображает сокращённый набор пунктов главного меню, и тогда пункт Forum среди них отсутствует. Чтобы он отобразился, надо попробовать развернуть окно веб-браузера на весь экран. Если это не поможет, то перейти на веб-страницу форума можно из раскрывающегося меню, как показано на рисунке 8.3.

Рис. 8.3

Переход на веб-страницу форума через раскрывающееся меню

Возможно, кому-то будет проще открыть веб-страницу форума путём набора её URL в адресной строке веб-браузера: http://fossil.server:8081/forum. Все перечисленные способы одинаково хороши и приводят к одному и тому же результату.

8.3. Начало обсуждения

Изначально форум пуст, о чём говорит сообщение на веб-странице: No forum posts found (На форуме отсутствуют сообщения). Поэтому кто-то должен выступить инициатором общения, предложив тему для обсуждения с помощью кнопки New Thread (Новая тема) (рис. 8.4).

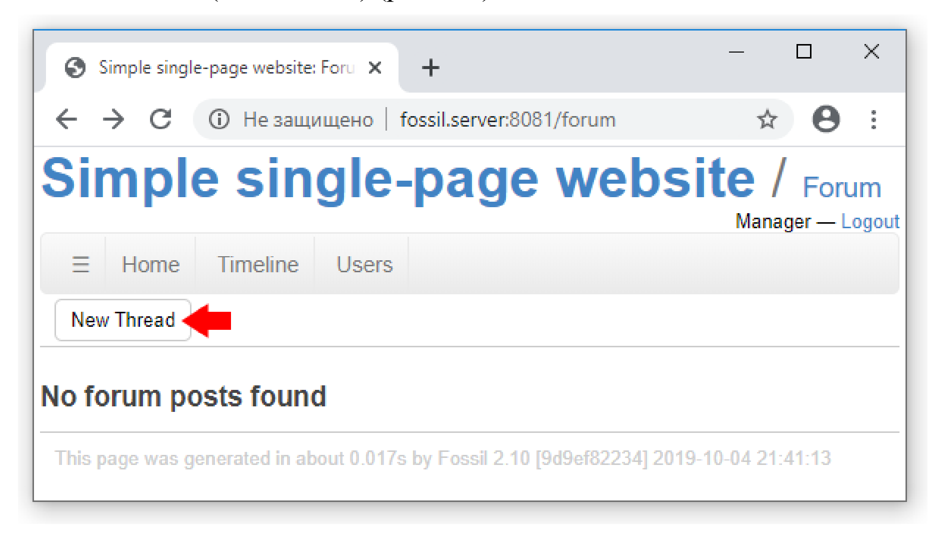


Рис. 8.4

Создание новой темы для обсуждения на форуме

После этого будет предложено заполнить форму, которая изображена на рисунке 8.5.

В поле From: (От:) указано имя пользователя, который создаёт тему. В поле Title: (Заголовок:) надо ввести заголовок темы, кратко отражающий круг обсуждаемых в ней вопросов.

В поле Markup style: (Способ разметки:) предлагается выбрать один из способов записи стилевого оформления, который будет использован при написании первого сообщения. Возможны следующие варианты:

- Fossil Wiki;
- Markdown;
- Plain Text.

Fossil Wiki — формат, в котором допускается использование многих тегов языка разметки HTML. Кроме того, абзацы можно обозначать пустыми строками, гиперссылки записывать в виде [URL|Описание], а элементы ненумерованных и нумерованных списков отмечать символами «звёздочка» * и «решётка» # соответственно, стоящими в начале строки и ограниченными с каждой стороны

двумя пробелами. Если в блоке текста требуется отключить форматирование, то его можно заключить в теги <nowiki> … </nowiki>.

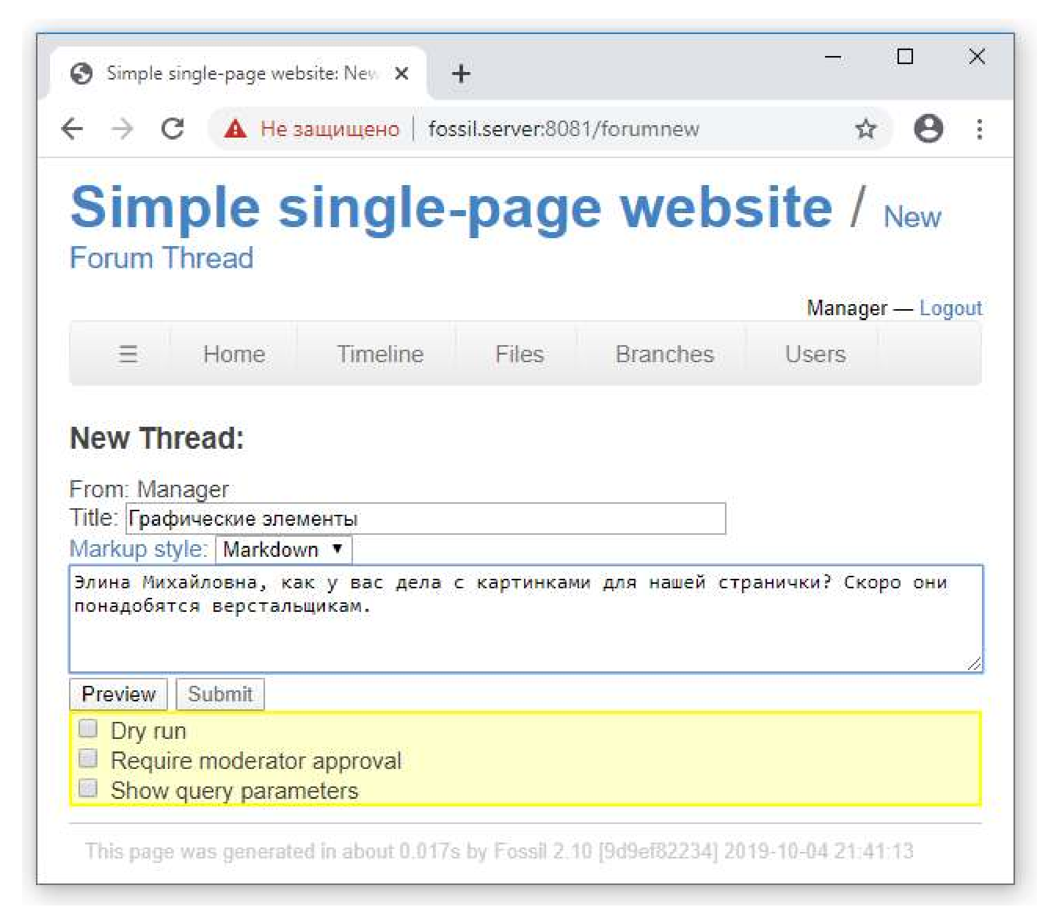


Рис. 8.5

Форма создания новой темы

Markdown — формат, в котором стилевое оформление вместо тегов задаётся специальными символами. Например, строка заголовка начинается с символов #, а количество этих символов означает уровень заголовка. Гиперссылки записываются в виде [Описание](URL). Для выделения слова курсивом его надо заключить в символы * или _, а для выделения жирным шрифтом — в символы ** или __.

Plain Text — это обычный текстовый формат, в котором никакие символы не несут специальных значений, а отображаются непосредственно в том виде, в котором они записаны.

Большое текстовое поле предназначено для написания первого сообщения в создаваемой теме. С помощью кнопки Preview (Предпросмотр) можно без записи сообщения в создаваемую тему форума посмотреть, как оно будет выглядеть после отправки. Это особенно полезно при использовании стилей оформ-

ления Fossil Wiki или Markdown, где некоторые комбинации символов могут быть интерпретированы неожиданным способом.

Чтобы создать ветку форума с заданной темой и набранным сообщением, надо нажать кнопку Submit (Представить), которая становится активной только после выполнения предварительного просмотра.

Если новую тему создаёт пользователь с правами администратора Fossil, то в нижней части формы отображаются три поля флажков. Два из них предназначены для отладки.

Установка флажка Show query parameters (Показать параметры запроса) приводит к тому, что после нажатия на кнопку Preview наряду с демонстрацией того, как будет показано сообщение после его отправки, выводятся значения параметров HTTP-запроса к веб-серверу Fossil (рис. 8.6).

Preview:

Графические элементы

Элина Михайловна, как у вас дела с картинками для нашей странички? Скоро они понадобятся верстальщикам.

```
CONTENT_LENGTH = 720
CONTENT_TYPE = application/x-www-form-urlencoded
GATEWAY_INTERFACE = CGI/1.0
HTTP_ACCEPT_ENCODING = gzip, deflate
HTTP_HOST = fossil.server:8081
HTTP_REFERER = http://fossil.server:8081/forumnew
HTTP_USER_AGENT = Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36
PATH_INFO = /forume1
QUERY_STRING =
REMOTE_ADDR = 192.168.56.1
REQUEST_METHOD = POST
REQUEST_URI = /forume1
SCRIPT_NAME =
content = Элина Михайловна, как у вас дела с картинками для нашей странички? Скоро они понадобятся верстальщикам.
mimetype = text/x-markdown
preview = Preview
showqp = on
title = Графические элементы
```

Рис. 8.6

Отображение параметров запроса
в режиме Show query parameters

Флажок Dry run (Пробный прогон) предотвращает добавление сообщения в тему форума по нажатию кнопки Submit. Вместо этого выводится описание того, какие изменения произвело бы добавление сообщения в системе Fossil (рис. 8.7).

This is the artifact that would have been generated:

```
D 2020-03-15T09:33:37.833
H Графические\sэлементы
N text/x-markdown
U Manager
W 189
Элина Михайловна, как у вас дела с картинками для нашей странички? Скоро они понадобятся верстальщикам.
Z 849e7f834c2a92dfb0b6f44d12bffb1a
```

Рис. 8.7

Отображение описания изменений
в режиме Dry run

Темы, которые создают пользователи с правами модератора форума, сразу становятся доступными для просмотра всем остальным участникам форума. Если требуется, чтобы даже созданная модератором тема ожидала проверки и одобрения публикации, то надо установить флажок Require moderator approval (Требуется одобрение модератора).

В результате подтверждения создания темы кнопкой Submit отобразится веб-страница с первым сообщением темы (рис. 8.8). Все участники форума под сообщением смогут увидель кнопку Reply (Ответить), с помощью которой можно написать ответное сообщение. Автору сообщения будут доступны ещё две кнопки: Edit (Редактировать) — для корректировки сообщения — и Delete (Удалить) — для его удаления.

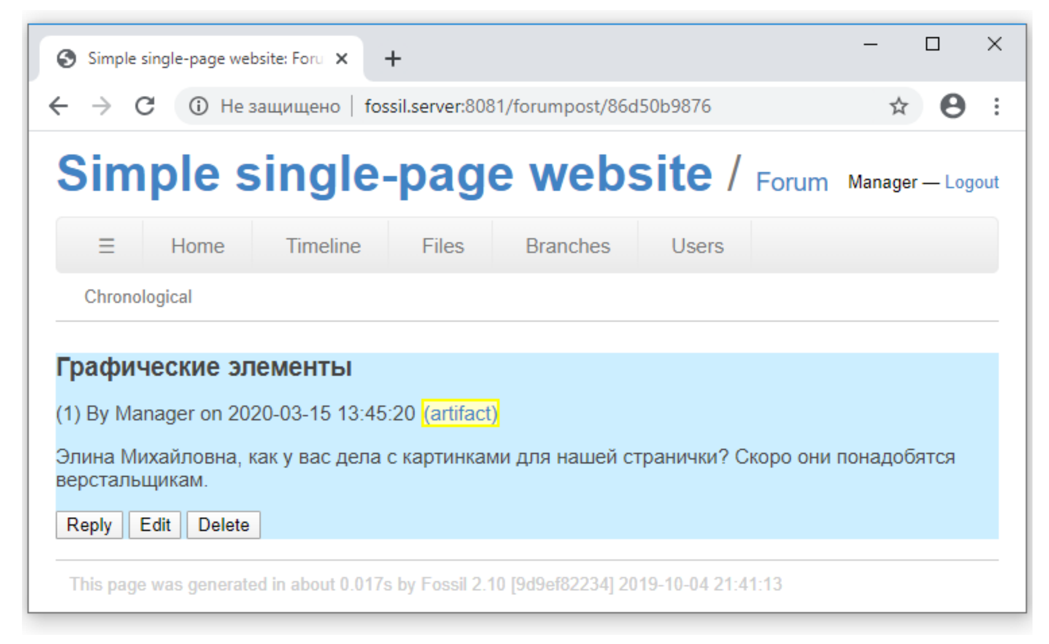


Рис. 8.8

Первое сообщение в новой теме

В этот же момент на главной веб-странице форума, которая ещё недавно была пустой, появится строка с созданной темой (рис. 8.9). Слева от названия темы отображается, сколько времени прошло с момента её создания. В дальнейшем эта отметка времени будет обнуляться при поступлении в тему новых сообщений, что позволяет быстро обнаружить в общем списке активно обсуждаемые темы. Фраза no replies (нет ответов) справа от названия означает, что ответы на опубликованное в теме сообщение ещё не поступали. В дальнейшем на её месте будет отображаться статистика поступления ответов.

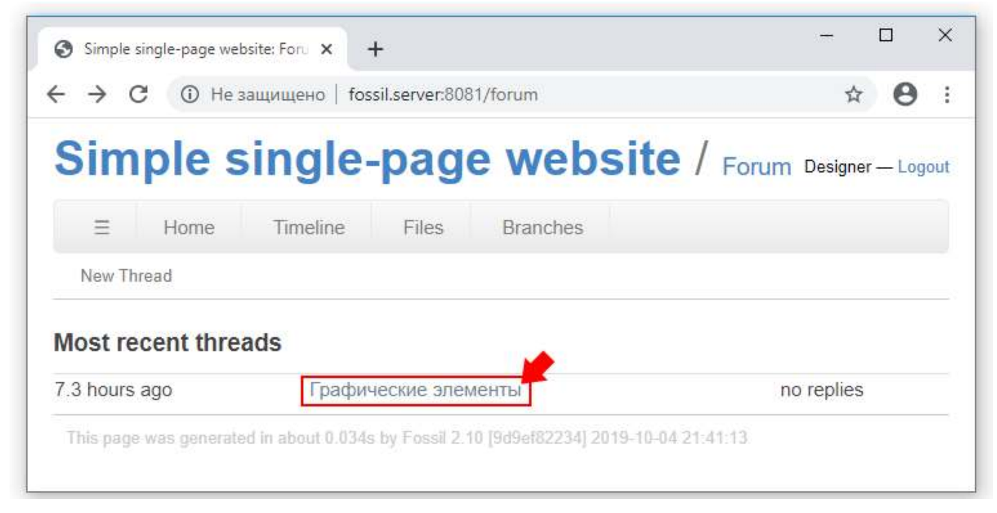


Рис. 8.9

Для просмотра обсуждений темы форума надо перейти по гиперссылке, представленной её названием

8.4. Участие в обсуждении

Чтобы просмотреть сообщения, опубликованные в теме, и, возможно, ответить на них, надо перейти по гиперссылке, которая представлена названием темы. После перехода откроется страничка, изображённая на рисунке 8.10. Пока что в теме одно сообщение — то, которое было набрано при создании темы.

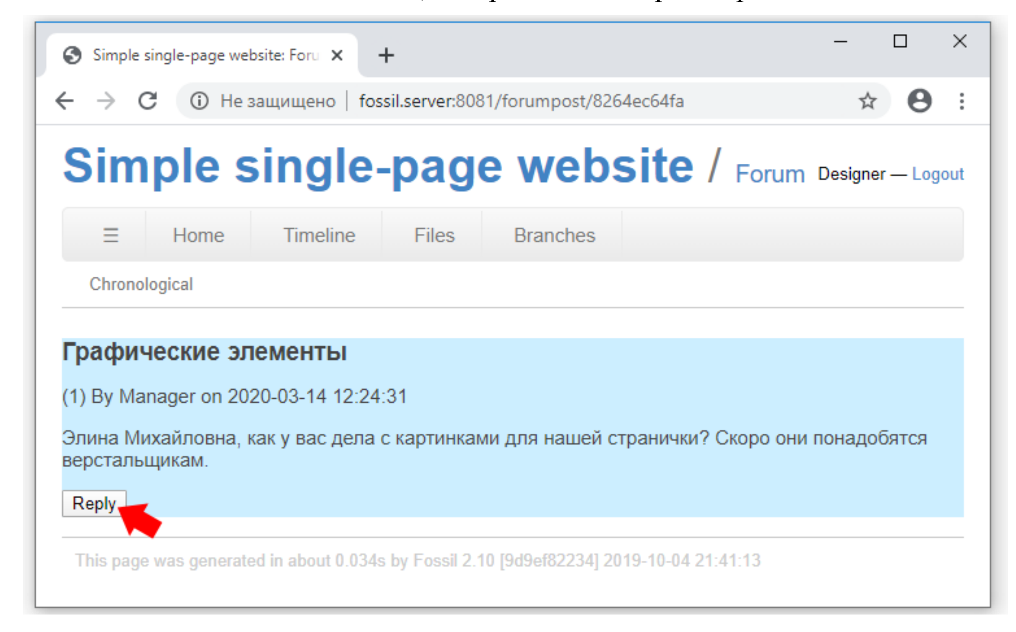


Рис. 8.10

Просмотр темы форума

Чтобы поддержать общение по теме, надо ответить на сообщение. Для этого предназначена кнопка Reply (Ответить). После нажатия на неё открывается форма, напоминающая форму создания темы (рис. 8.11). Только поле заголовка в ней уже заполнено, теперь достаточно набрать ответ в основном поле ввода текста. Единственный нюанс — сначала в нижней части формы присутствуют лишь две кнопки: Preview (Предпросмотр) и Cancel (Отмена). Кнопка для отправки Submit (Представить) набранного сообщения появится только после того, как оно будет просмотрено с помощью кнопки Preview.

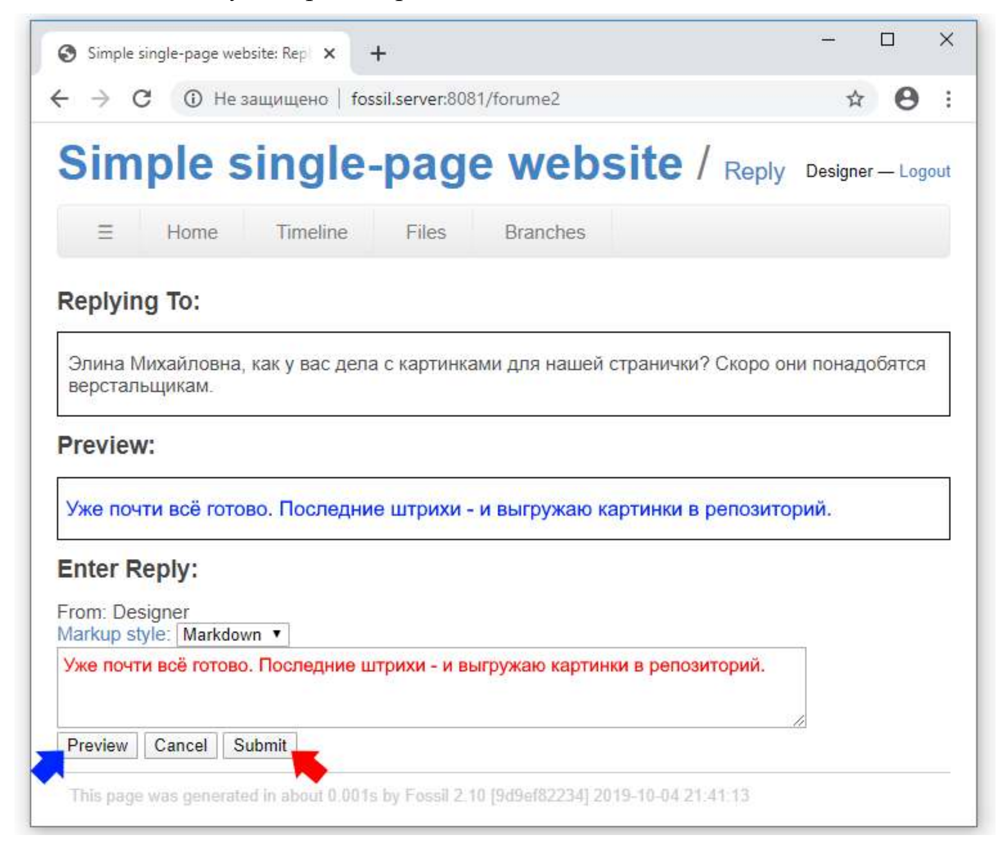


Рис. 8.11

Подготовка ответа на сообщение форума

Сообщение, подтверждённое кнопкой Submit, попадает в ветку форума, посвящённую текущей теме (рис. 8.12). Несмотря на то, что автор сообщения видит под ним кнопку Delete, удалить сообщение с её помощью он не сможет. При нажатии на неё открывается форма ответа на созданное сообщение. Скорее всего, это временная проблема, которая будет исправлена в следующих версиях Fossil. Однако у неё есть корни, о которых стоит упомянуть.

В связи с тем, что форум в системе Fossil использует те же механизмы, на которых работает система контроля версий, процедура удаления сообщений из форума не может происходить обычным образом. Дело в том, что репозиторий — это хранилище, в которое могут только добавляться новые объекты и из

которого никогда ничего не удаляется. Поэтому удалить сообщение, отправленное на форум, нельзя. Можно лишь изменить его на какое-нибудь другое сообщение, которое перекроет старое (хотя в истории Fossil старое сообщение всё равно сохранится). Это цена, которая вполне оправдана компактностью системы. Реализация полноценного форумного механизма в рамках Fossil не имеет смысла — для этого есть специализированные решения.

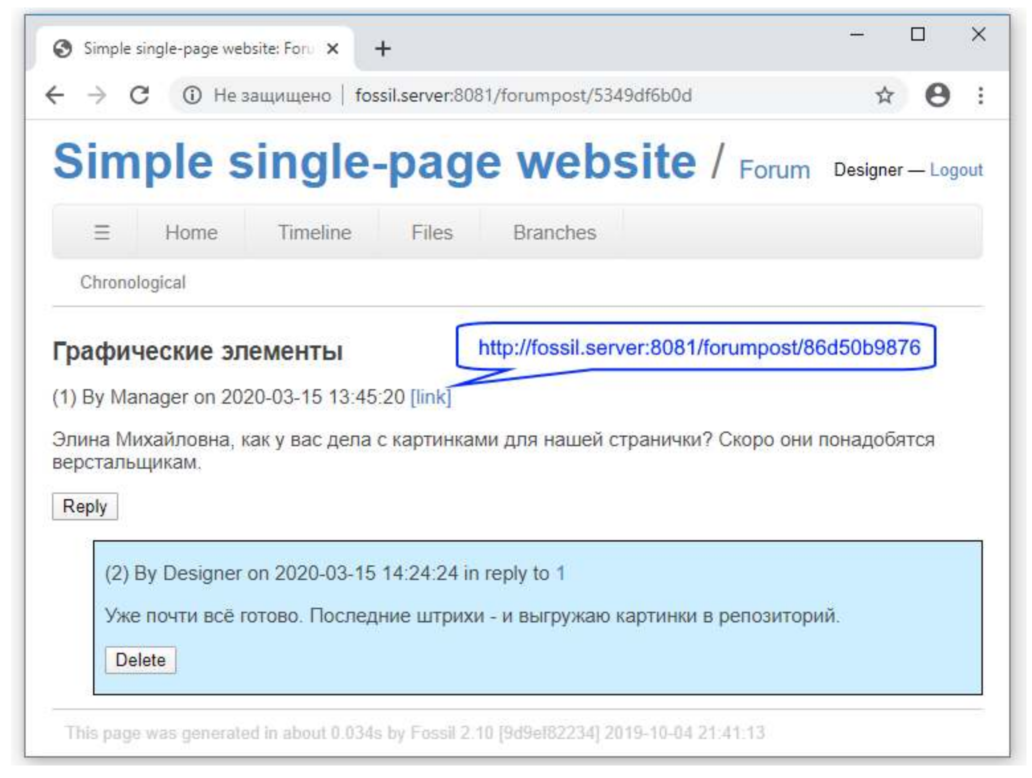


Рис. 8.12

Тема форума после опубликования ответа. Гиперссылка [link] позволяет обращаться к сообщению из других тем или подсистем Fossil

Стоит обратить внимание на гиперссылку [link], которая находится в заголовке сообщения. Она представляет собой постоянный адрес этого сообщения на форуме. Её можно копировать в буфер обмена с тем, чтобы потом использовать для ссылки на сообщение из других информационных ресурсов системы Fossil, например при написании заявки на доработку. Такие, казалось бы, мелочи обогащают информационное наполнение проекта, упрощают донесение своих мыслей до других участников и включают всю накопленную информацию в активное пользование. При щелчке по этой гиперссылке на форуме соответствующее ей сообщение становится активным, т. е. выделяется фоновым цветом.

Теперь посмотрим, как выглядит обсуждение с точки зрения пока ещё стороннего наблюдателя — верстальщика Imposer1. Вот что он увидит, если зайдёт прямо сейчас на страницу, посвящённую обсуждаемой теме (рис. 8.13).

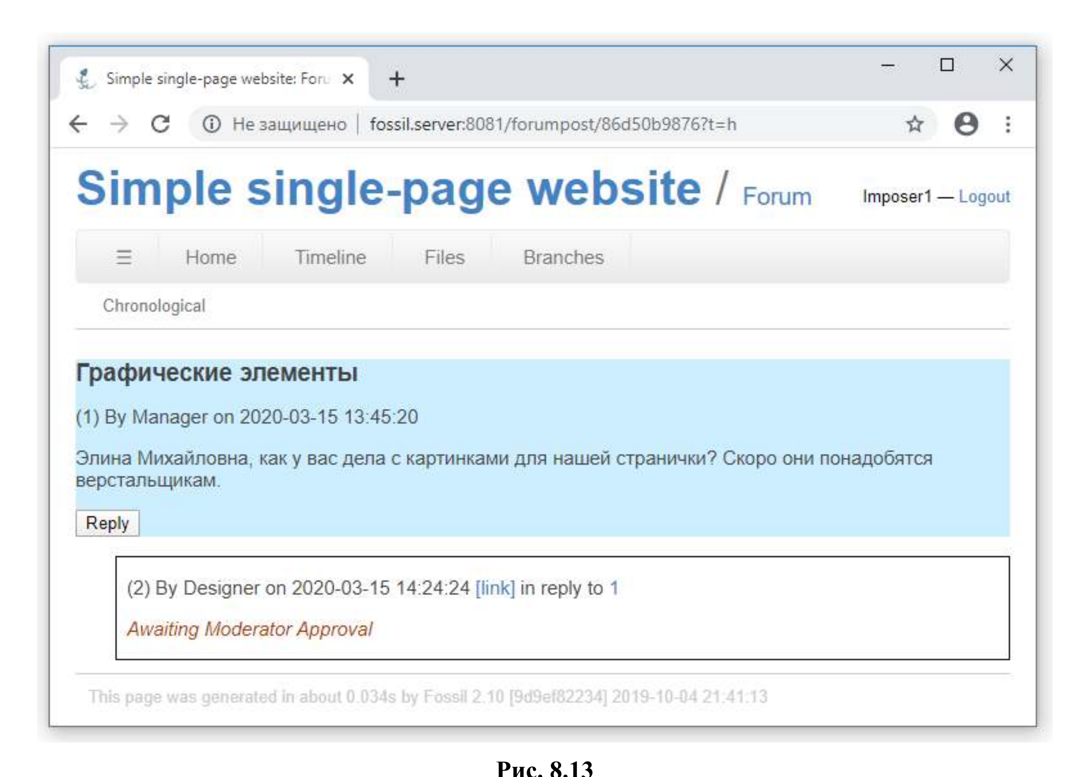


Рис. 8.13

Сообщения на форуме в ожидании модерирования

Ответ, который был написан дизайнером, обозначен на форуме блоком, состоящим из заголовка (в котором указано имя автора и время создания сообщения) и фразы Awaiting Moderator Approval (Ожидает одобрения модератором), которая заменяет оригинальный текст сообщения. Так работает механизм защиты форума от спама: сообщения от рядовых пользователей, не являющихся модераторами, не отображаются до тех пор, пока не будут проверены и одобрены одним из модераторов.

8.5. Действия модератора

Единственный пользователь, который имеет права модератора в рассматриваемом примере, — это администратор системы, менеджер (Manager). Вот как для него выглядит то же самое обсуждение (рис. 8.14).

Во-первых, модератор видит всё сообщение целиком, вместе с его оригинальным текстом. Во-вторых, после ознакомления с текстом сообщения модератор может опубликовать его так, чтобы оно стало видно всем посетителям форума. Это выполняется с помощью кнопки Approve (Одобрить). Если же текст сообщения не удовлетворяет пользовательским соглашениям, то с помощью кнопки Reject (Отклонить) модератор может отказать в его публикации. В последнем случае сообщение скрывается от всех пользователей.

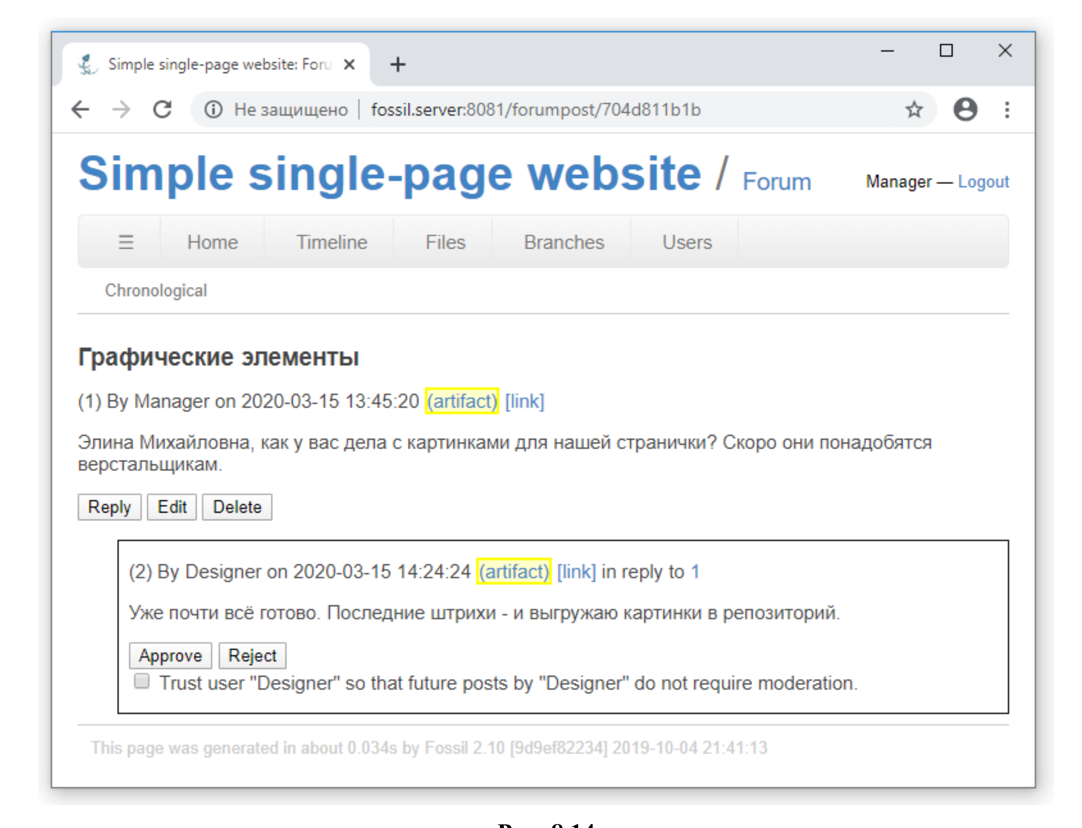


Рис. 8.14

Вид сообщений с точки зрения модератора

Если пользователь, опубликовавший сообщение, заслуживает доверия, то модератор может перед одобрением установить флажок Trust user ... so that future posts do not require moderation. Тогда в дальнейшем сообщения, отправляемые этим пользователем, будут сразу опубликовываться для всех участников форума, не требуя одобрения модератора.

Как и любой другой пользователь форума, модератор может принять участие в обсуждении. Такую возможность ему предоставляет кнопка Reply для ответа на сообщение. Процедура написания ответа уже была рассмотрена выше. В отличие от рядовых пользователей, модератор может осуществлять над опубликованными сообщениями дополнительные действия.

Во-первых, с помощью кнопки Edit он может внести изменение в сообщение любого пользователя. Процедура редактирования работает точно так же, как и ввод нового сообщения или ответ на существующее, за тем лишь исключением, что поле для ввода текста сообщения не пустое, а уже содержит текст исправляемого сообщения. Сохранение внесённых изменений производится последовательным нажатием кнопок Preview и Submit.

Во-вторых, модератор может удалить сообщение любого пользователя. Для этого предназначена кнопка Delete (Удалить). После нажатия на неё открывается страница с запросом подтверждения (рис. 8.15).

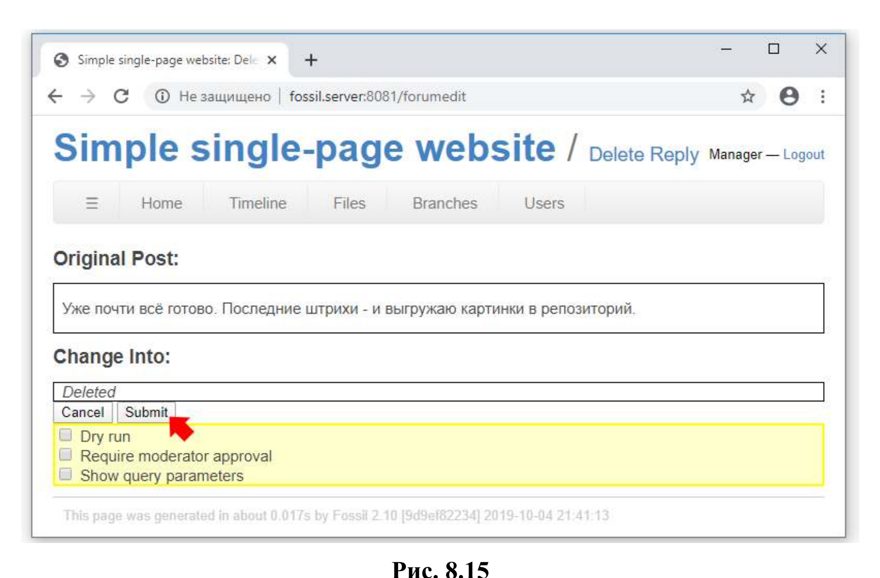


Рис. 8.15

Запрос на подтверждение удаления сообщения

Для окончательного удаления сообщения надо подтвердить принятое решение с помощью кнопки Submit (Подтвердить). Если же кнопка Delete была нажата по ошибке, то можно отказаться от удаления с помощью кнопки Cancel (Отменить). Удалённое таким образом сообщение не исчезает из обсуждения, а остаётся в нём на прежнем месте (рис. 8.16), только текст сообщения заменяется на фразу Deleted (Удалено).

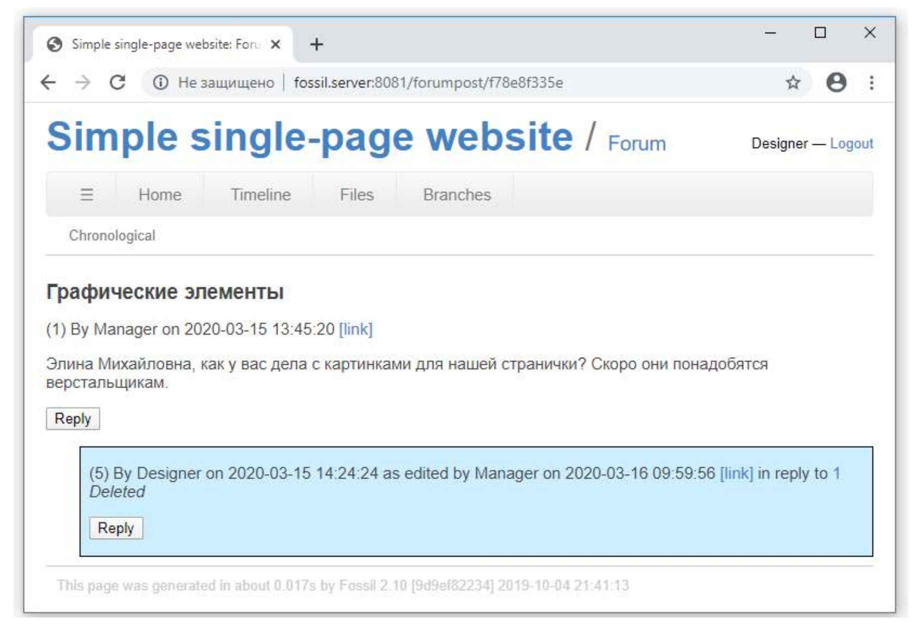


Рис. 8.16

Удалённое сообщение остаётся в обсуждении на прежнем месте, только вместо его текста отображается Deleted

При эксплуатации системы Fossil сообщения будут поступать в разные темы обсуждений, и модератор может упустить из вида некоторые из них, в результате чего общение остановится. Чтобы увидеть сразу все сообщения, требующие его внимания, модератор может воспользоваться пунктом раскрывающегося меню Administration Pages | Pending Moderation Requests (рис. 8.17).

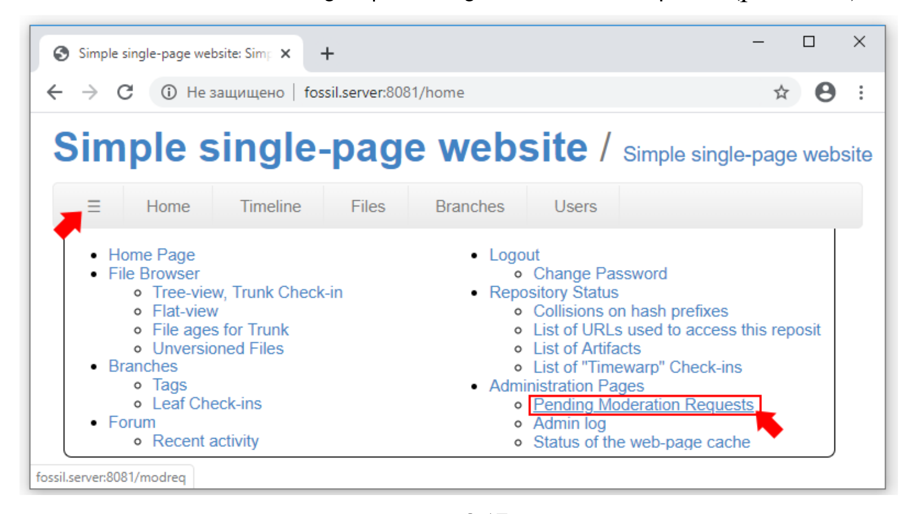


Рис. 8.17

Переход к списку сообщений, ожидающих проверки модератором

В появившемся списке легко перебрать все сообщения, просматривая их по гиперссылкам с идентификаторами (рис. 8.18) и одобряя или запрещая их публикацию.

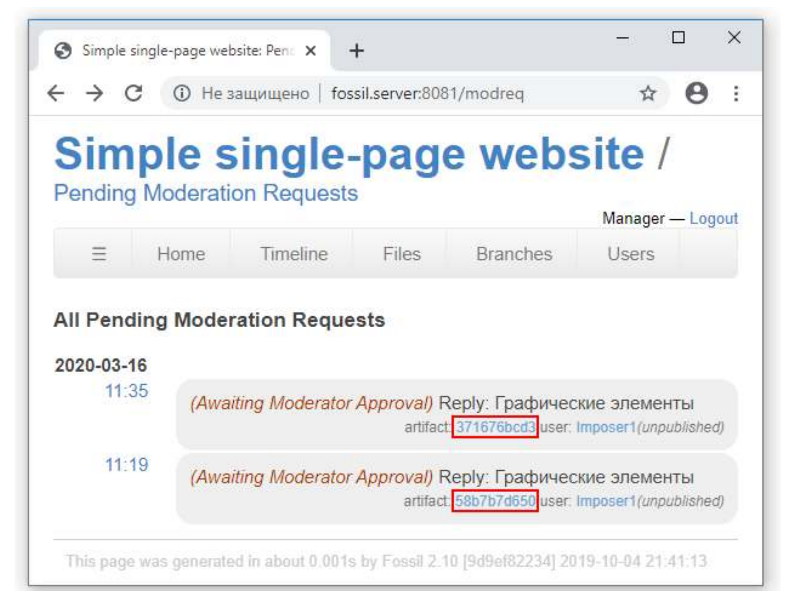


Рис. 8.18

Список сообщений, ожидающих проверки модератором

Глава 9
СЕРВЕР

9.1. Протокол передачи гипертекста

Соединение компьютеров кабелем или с помощью волновых приёмо-передатчиков для обеспечения информационного обмена — важный, но далеко не решающий шаг в расширении сфер их применения. Для использования этой технической возможности требуется программное обеспечение, которое будет формировать информационные блоки для передачи и обрабатывать их при получении, реализуя при этом какую-то полезную функцию, например, резервное копирование файлов или распространение новостных сообщений.

Совокупность правил и соглашений, описывающих структуру блоков информации, которые участвуют в информационном обмене, и порядок (регламент) такого обмена называются протоколом. Например, для передачи простых наборов данных между рядом стоящими компьютерами в локальной сети предназначен протокол Ethernet, для обмена такими наборами в сложной сети с промежуточными станциями — маршрутизаторами — обычно используют протокол IP. Чтобы адресовать пакет информации конкретной программе, выполняющейся на компьютере, применяют протокол UDP, а для установления канала передачи данных между программами, выполняющимися на различных компьютерах, — протокол TCP.

Перечисленные протоколы решают сложную и важную задачу доставки блоков информации с одного компьютера на другой через не всегда надёжные линии связи, но их затруднительно использовать в прикладных целях. Например, чтобы записать файл на сервер, недостаточно передать содержащиеся в этом файле данные с одного компьютера на другой. Надо сначала определить сетевой адрес сервера по его имени, затем проверить, имеет ли пользователь право записи на сервер, а потом вместе с содержимым файла передать (и обработать) его атрибуты. При использовании протоколов UDP или TCP для этого потребуется много отдельных информационных обменов, что непросто реализовать в программах, нуждающихся в передаче файлов. Поэтому были разработаны протоколы более высокого уровня, например FTP, NFS и SMB/CIFS, использующие протоколы UDP и TCP, но скрывающие сложность решения этой задачи.

В конце 1980-х гг., когда сеть Интернет в виде обмена данными между компьютерами уже состоялась (например, в ней функционировали серверы обмена файлами FTP и передачи электронной почты SMTP/POP3), Тим Бернерс Ли предложил протокол для размещения в этой сети гипертекстовой информации — текстовых документов, связанных между собой перекрёстными ссылками.

В результате в 1991 г. возникла всемирная паутина World Wide Web, ныне известная под аббревиатурой WWW, которая базируется на двух китах — языке гипертекстовой разметки HTML (HyperText Markup Language), на котором пи-

шутся текстовые документы для размещения в WWW, и протоколе HTTP (HyperText Transfer Protocol), предназначенном для обмена этими документами в сети Интернет. Протокол HTTP как соглашение о правилах информационного обмена используется двумя типами программ — веб-серверами, которые работают на компьютерах, хранящих гипертекстовые документы, и веб-браузерами, которые выполняются на компьютерах пользователей, запрашивают необходимые документы у серверов и отображают их содержимое на экранах. При передаче данных, как уже было сказано выше, неявно используются низкоуровневые протоколы TCP, IP, Ethernet, WiFi (рис. 9.1).

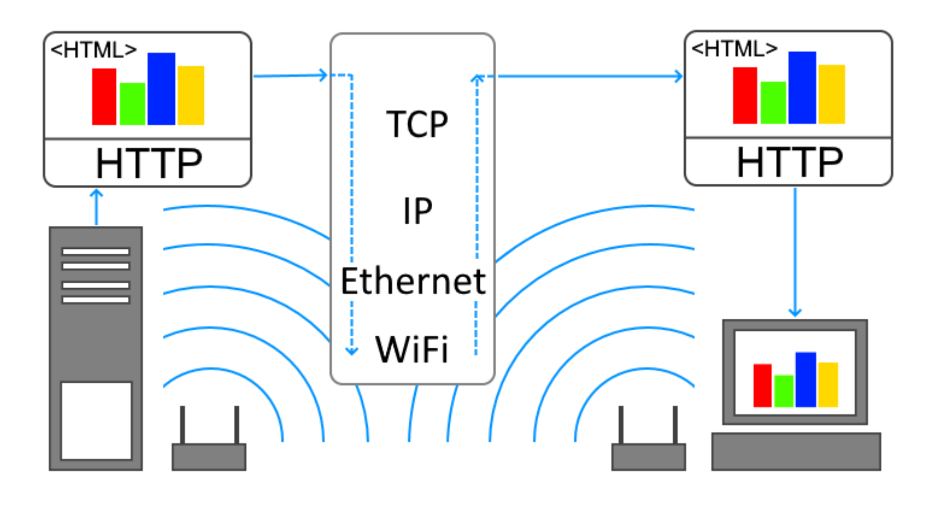


Рис. 9.1

Передача гипертекстового документа от сервера к клиенту по протоколу HTTP. При этом неявно используются другие, более низкоуровневые протоколы

Если низкоуровневые протоколы, поскольку они универсальны и требуются многим программам, реализуются модулями, входящими в состав операционной системы, то высокоуровневые протоколы, предназначенные для решения конкретных задач, реализуются теми прикладными программами, которым они необходимы. Оформление данных в соответствии с протоколом HTTP выполняется веб-сервером, а их интерпретация и распаковка — веб-браузером.

Соглашения, входящие в состав протокола HTTP, описывают не только способы доставки документов от сервера к клиенту, но и передачу информации в обратном направлении — от клиента к серверу. Это позволяет реализовывать полноценные веб-приложения, принимающие и обрабатывающие пользовательский ввод и выдающие ответы на запросы пользователей. В полной мере возможности протокола реализуют такие специализированные веб-серверы, как Apache и nginx. В дополнение к базовой функциональности они умеют производить авторизацию пользователей и формировать гипертекстовые документы на лету по программам, написанным на специальных языках программирования, с использованием информации, хранящейся в базах данных.

Однако богатые возможности, заложенные в протокол HTTP, в полной мере используются далеко не всегда. Во многих случаях от веб-сервера требуется лишь выдать в ответ на запрос пользователя гипертекстовый документ, хранящийся в HTML-файле. Этого достаточно для реализации статического веб-сайта, раздающего редко изменяющуюся информацию. В таких случаях установка, а главное, настройка полноценного веб-сервера может быть совершенно лишней. Такую простую функциональность умеют реализовывать многие программы, в число которых входит и Fossil.

9.2. Язык разметки гипертекста

Чтобы веб-браузеры могли правильно отображать содержимое гипертекстовых документов и корректно обрабатывать содержащиеся в них гиперссылки, текст документов должен быть оформлен в соответствии с правилами языка разметки гипертекста HTML. В настоящий момент актуальной является пятая версия этого языка разметки. Структура правильно оформленного HTML-документа выглядит следующим образом:

Структура HTML-документа

```
<!doctype html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Шаблон HTML-документа</title>
  </head>
  <body>
    <h1>Шаблон HTML-документа</h1>
    <section>
      <h2>Первая секция</h2>
      <p>Содержимое первой секции.</p>
    </section>
    <section>
      <h2>Первая секция</h2>
      <p>Содержимое первой секции.</p>
    </section>
  </body>
</html>
```

Документ, соответствующий требованиям HTML5, — это текстовый файл, обычно имеющий расширение .html, в котором наряду с основным информационным наполнением содержатся теги — специальные обозначения в угловых скобках, описывающие структуру документа. В настоящее время HTML-документы обычно сохраняются в кодировке UTF-8 (как правило, кодировку можно указать при сохранении текста в редакторе). Будем исходить именно из этого предположения, потому что в случае неправильной кодировки при просмотре документа в веб-браузере текст на русском языке может стать нечитаемым.

Первой в HTML-документе должна идти строка, декларирующая его тип: <!doctype html>. Она информирует веб-браузер о том, что в дальнейшем содержимое документа следует трактовать в соответствии с правилами HTML5.

Собственно HTML-документ заключён в теги <html> и </html> и состоит из двух частей — заголовка, заключённого в теги <head> и </head>, и тела, заключённого в теги <body> и </body>. В открывающем теге <html> должен присутствовать атрибут *lang*, указывающий, на каком языке написано содержимое документа. В рассматриваемом примере значение ru этого атрибута указывает на русский язык. Участок документа, состоящий из открывающего и закрывающего тегов вместе с находящимся между ними содержимым, называется элементом HTML.

В заголовке документа обычно указывается служебная информация. В элементе <title> записывается название документа, которое отображается на вкладке веб-браузера и используется в качестве одного из основных ориентиров поисковыми системами Интернет, такими как Яндекс или Google. С помощью атрибута charset тега <meta> указывается, в какой кодировке представлен текст документа. Это важно для правильного отображения содержимого документа в веб-браузере.

В теле документа находится его основное содержимое — та информация, которая должна быть отображена в веб-браузере и представлена пользователю. Обычно в самом начале тела документа находится заголовок верхнего уровня <h1>, относящийся ко всему документу в целом. Информационное наполнение может быть разбито на несколько секций с помощью тегов <section> и </section>, внутри которых текст делится на абзацы с помощью тегов <p> и </p>. В соответствии со стандартом HTML5 каждая секция должна иметь свой заголовок — в приведенном примере это заголовки второго уровня, представленные элементами <h2>.

Деление содержимого документа на секции не является обязательным, оно может быть представлено в теле документа непосредственно. Помимо секций в языке HTML имеются и другие возможности для организации содержимого в блоки с учётом его смысловой нагрузки (заголовки, подвалы) или без её учёта (подразделы).

Начальные сведения о том, как с помощью HTML изменять начертание текста, вставлять графические иллюстрации и создавать гиперссылки, приведены в разделе о системе документирования. Языку HTML посвящено множество учебников, в том числе информационно-справочных ресурсов в сети Интернет, поэтому найти интересующую информацию не составит труда, а представленных в этой книге сведений достаточно для понимания дальнейшего материала.

9.3. Веб-сервер

Для реализации дополнительной функции статического веб-сервера в Fossil разработчикам не пришлось прилагать много усилий. Система Fossil изначально была ориентирована на использование веб-интерфейса в качестве од-

ного из основных средств управления репозиторием проекта и поэтому обязана уметь обрабатывать протокол HTTP. Возможность выдачи по этому протоколу HTML-документов, содержащихся в отдельных файлах, стала приятным расширением уже реализованной функциональности.

Ранее был описан способ запуска Fossil для предоставления доступа к сетевому репозиторию, хранящемуся на сервере в файле sspwebsite.fossil, с помощью команды:

```
fossil server -P 8081 sspwebsite.fossil
```

Но если в организации ведётся разработка нескольких проектов, то естественной является потребность одновременной работы с несколькими репозиториями. В этом случае файлы репозиториев можно собрать в одном каталоге, который называется, например, /var/repos, а затем предоставить к ним сетевой доступ командой:

```
fossil server -P 8081 /var/repos
```

После этого репозитории, хранящиеся в файлах rep1.fossil и rep2.fossil, станут доступны по гиперссылкам http://fossil.server:8081/rep1 и http://fossil.server:8081/rep2 соответственно, в которых fossil.server — это сетевой адрес сервера Fossil. Таким образом, один сервер может обслуживать любое количество репозиториев.

Но что произойдёт, если попытаться обратиться к такому многорепозиторному серверу по гиперссылке http://fossil.server:8081/, без указания имени конкретного репозитория? В этом случае сервер Fossil ответит «Not found» и вернёт ошибку 404 — запрошенный ресурс не найден. Так отвечают веб-серверы, когда к ним обращаются по неправильным гиперссылкам.

Можно сделать так, чтобы при обращении к серверу Fossil без указания имени репозитория пользователю возвращался список репозиториев, обслуживаемых сервером, с возможностью выбора одного из них для дальнейшей работы (рис. 9.2). Для этого достаточно добавить параметр *--repolist* в команду запуска сервера:

```
fossil server -P 8081 --repolist /var/repos/
```

Это, безусловно, полезная функция, но от веб-сервера ожидают большего. Как минимум он должен возвращать по запросу файлы, составляющие веб-страницу. Обычно речь идёт о файлах, имеющих расширение:

- .html — содержимое веб-страницы, оформленное на языке разметки HTML;
- .css — таблицы стилей, записанные на языке CSS;
- .js — исходные тексты сценариев на языке JavaScript;
- .gif, .jpg, .png — файлы с изображениями для иллюстрирования веб-страницы.

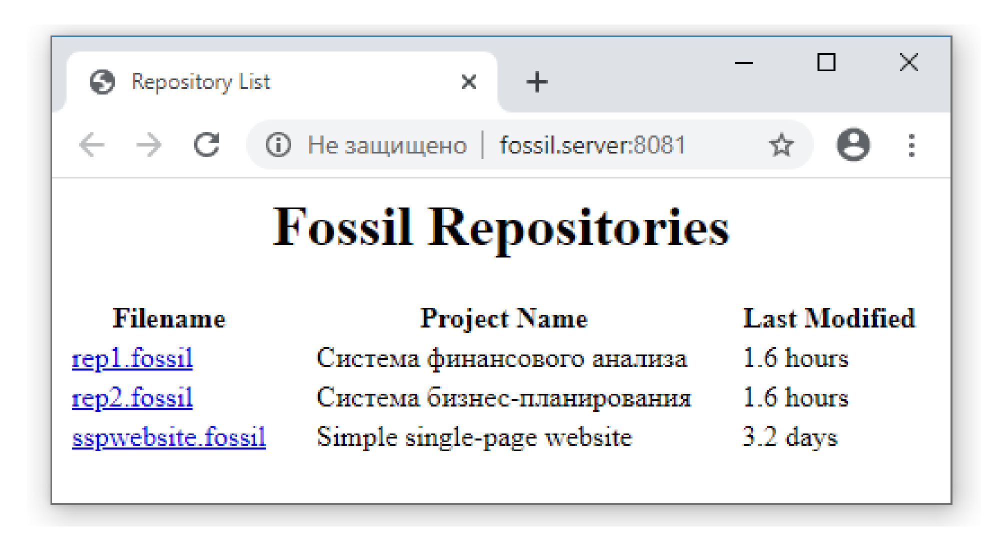


Рис. 9.2

Сервер Fossil умеет формировать список обслуживаемых им репозиториев

В простом текстовом редакторе наберём текст HTML-странички и сохраним его в файле index.html в кодировке UTF-8 (обычно текстовый редактор предлагает выбрать кодировку при сохранении):

Файл index.html для стартовой страницы сервера Fossil

```
<!doctype html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Сервер Fossil</title>
</head>
<body>
<h1>Корпоративный сервер Fossil</h1>
<h2>Обслуживаемые репозитории:</h2>
<ul>
<li><a href="/sspwebsite">Одностраничный веб-сайт</a></li>
<li><a href="/rep1">Система бизнес-планирования</a></li>
<li><a href="/rep2">Система финансового анализа</a></li>
</ul>
</body>
</html>
```

Поместим созданный файл index.html в каталог /var/repos, где уже находятся файлы rep1.fossil и rep2.fossil, после чего запустим веб-браузер и наберём в адресной строке: http://fossil.server:8081/index.html. Получим ту же самую ошибку Not found, сообщающую о недоступности запрошенного файла.

Дело в том, что по умолчанию веб-сервер Fossil настроен на обслуживание исключительно файлов с расширением .fossil, в которых хранятся репозито-

рии. Чтобы он стал замечать файлы с другими расширениями, надо в команде запуска сервера перечислить шаблоны имён этих файлов в параметре *--files*:

```
fossil server -P 8081 --files *.html,*.css,*.js,*.jpg,*.gif,*.png /var/repos/
```

После этого веб-сервер станет отвечать по указанному выше адресу созданной веб-страницей (рис. 9.3).

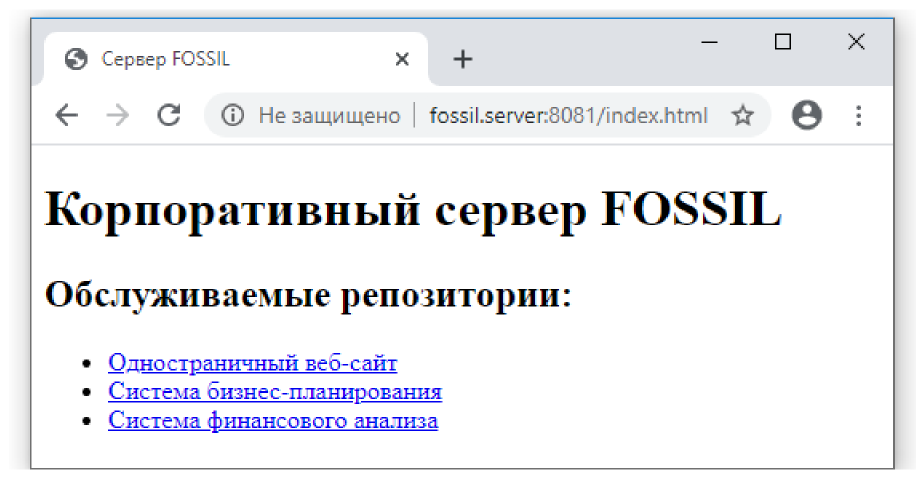


Рис. 9.3

Веб-сервер возвращает файл index.html, расположенный в каталоге вместе с файлами репозиториев

Можно пойти дальше и создать на сервере в каталоге /var/repos подкаталог website, который сделать рабочим каталогом для разрабатываемого веб-сайта:

```
mkdir /var/repos/website
cd /var/repos/website
fossil open ../sspwebsite.fossil "основная"
```

После этого по адресу http://fossil.server:8081/website/index.html будет открываться веб-страничка, разрабатываемая в рамках проекта «Простой одностраничный веб-сайт». Необходимость указания в последней команде названия ветви «основная» возникла по той причине, что главная ветвь разработки, по умолчанию называемая trunk, была переименована в приведенном ранее примере.

С одной стороны, это позволяет разработчикам оценить разрабатываемый веб-сайт в действии. С другой, система Fossil начинает выступать в роли системы доставки содержимого на сервер и своеобразной системы управления содержимым — CMS. Осталось на сервере организовать периодическое выполнение команды обновления рабочего каталога (например, с помощью входящего в операционную систему планировщика):

```
fossil checkout --latest --force
```

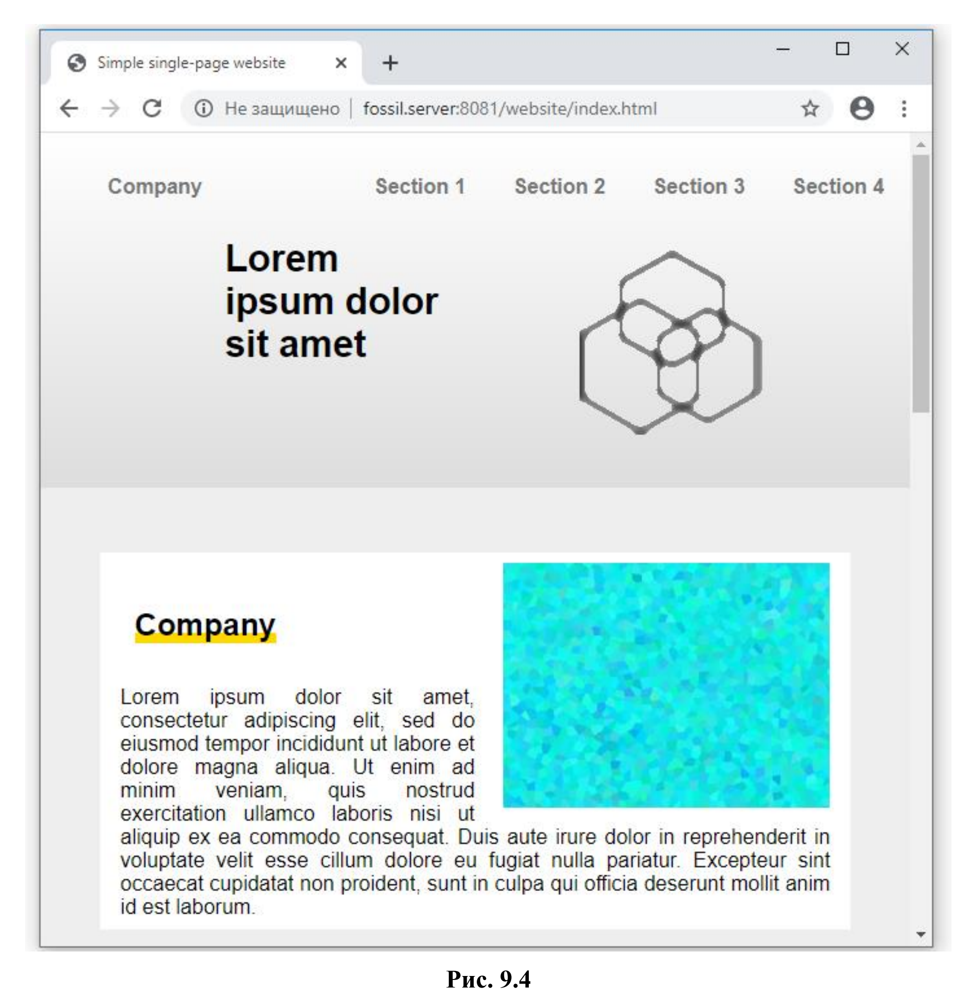


Рис. 9.4

Сервер Fossil предоставляет доступ к веб-сайту, построенному из содержимого репозитория одного из проектов

Тогда в рабочем каталоге веб-сайта на сервере будут находиться автоматически обновляемые версии файлов проекта, куда будут попадать все вносимые разработчиками и отправляемые в репозиторий изменения. Такой вариант использования может представлять интерес для организации функционирования корпоративного веб-сайта. Помимо регулярных обновлений он позволяет в любой момент восстановить любую историческую версию содержимого сайта из своего репозитория.

Если зайти на официальный сайт системы Fossil https://fossil-scm.org/, который работает под управлением этой же системы (самой актуальной её версии), то можно заметить его более богатое оформление (рис. 9.5) по сравнению с веб-интерфейсом репозитория, рассматриваемого в этой книге. Прежде всего бросаются в глаза информативная стартовая страница Home и изменённое главное меню, которое содержит нестандартные пункты: Docs — для перехода к странице сопроводительной документации, Code — для просмотра последних версий

файлов проекта, Support — для перехода на страницу форума — и Download — для скачивания дистрибутивов системы Fossil.

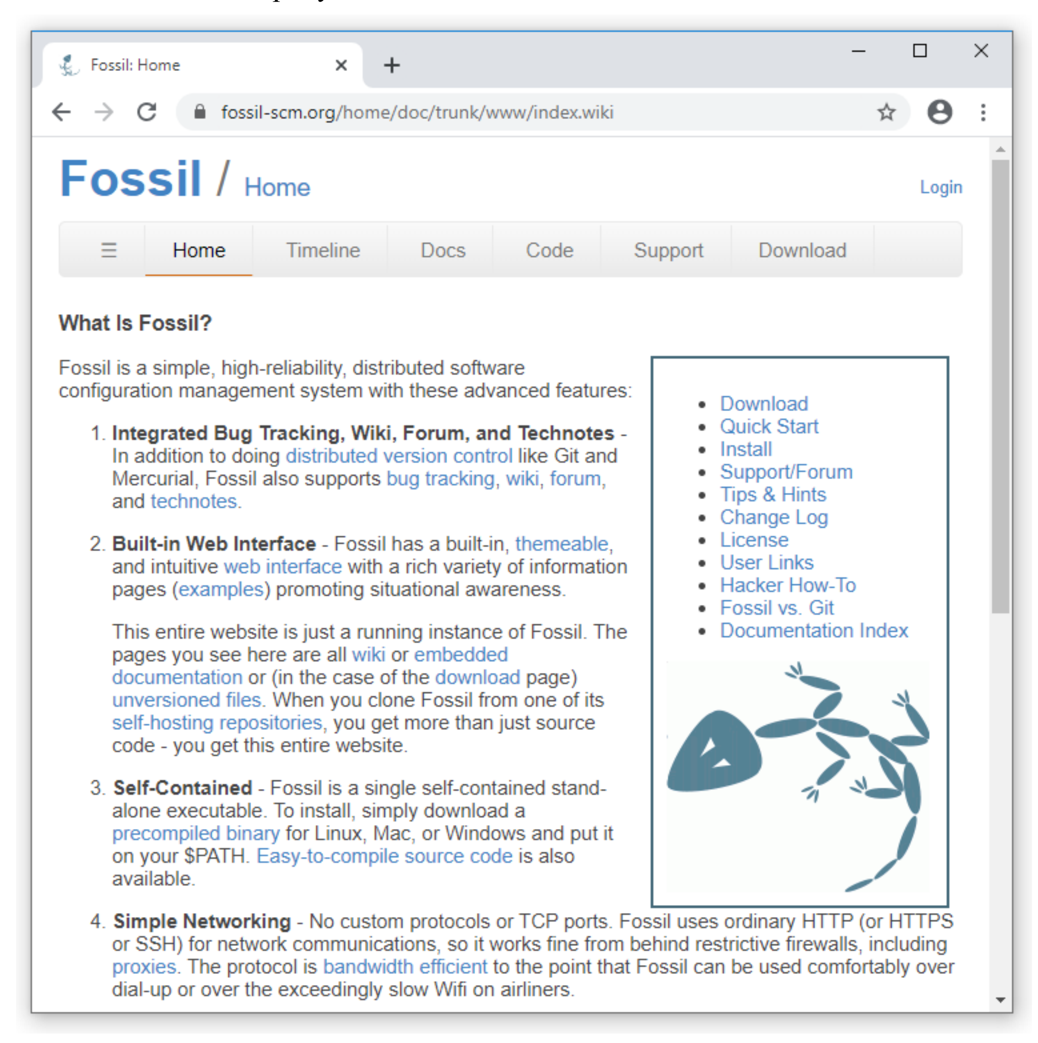


Рис. 9.5

Главная страница официального веб-сайта проекта Fossil

Попытаемся улучшить веб-интерфейс нашего репозитория.

9.4. Встроенная документация

Ранее была рассмотрена система документирования Fossil, которая позволяет вести информационные странички со статьями, посвящёнными различным аспектам разрабатываемого проекта и оформленные с помощью языков разметки Wiki или Markdown. Помимо этого Fossil даёт возможность использовать в качестве страниц документации файлы, являющиеся частью проекта и находя-

щиеся в его рабочем каталоге. Такой вариант документирования имеет следующие преимущества:

- версия документации всегда соответствует версии разрабатываемого проекта и может быть выгружена вместе с его исходными текстами в одном архиве;
- для редактирования файлов, содержащих страницы документации, можно использовать любой текстовый редактор или привычную среду разработки;
- вносить изменения в документацию могут только участники проекта, имеющие права разработчиков (иногда это может быть и недостатком).

9.4.1. Работа над документацией

Вернёмся к примеру разработки шаблона одностраничного веб-сайта. Предположим, что работу над документацией менеджер решил вести в отдельной ветви проекта. Поскольку для неё не требуются никакие наработки из исходных файлов проекта, в качестве основы этой ветви менеджер решил использовать начальную точку сохранения репозитория, соответствующую пустому моментальному снимку рабочего каталога. Идентификатор этой точки сохранения — 24e23c5cfb, поэтому команда создания ветви «документация» приобрела такой вид:

```
fossil branch new "документация" 24e23c5cfb
```

Команда завершилась без ошибок:

```
New branch:
30ccf6ad616630ffa9ce8a97650d487e0fa2d6b6fe2e42a346f8d59b7d3c396a
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 1 received: 0
Push done, sent: 909 received: 281 ip: 192.168.56.101
```

Для файлов с документацией менеджер создал отдельный пустой каталог document (за пределами рабочего каталога с файлами исходных текстов шаблона одностраничного веб-сайта), сделал его текущим и открыл из него только что созданную ветвь «документация». Всё перечисленное он выполнил с помощью последовательности команд:

```
cd ..
mkdir document
cd document
fossil open ..\sspwebsite-local.fossil "документация"
```

Таким образом, у менеджера появился рабочий каталог, ассоциированный с ветвью «документация»:

```
project-name: Simple single-page website
repository: …/Manager/document/..\sspwebsite-local.fossil
local-root: …/Manager/document/
config-db: C:/Users/PCUser/AppData/Local/_fossil
project-code: 53e4e1445fb2a7289eb1a337ce1cf8c9a99f7b86
```

```
checkout: 30ccf6ad616630ffa9ce8a97650d487e0fa2d6b6 2020-05-22
18:11:38 UTC
parent: 24e23c5cfbbbd7be781d6a8b6e7adc8e12e9cdc4 2020-02-29
07:24:50 UTC
tags: документация
comment: Create new branch named "документация" (user: Manager)
check-ins: 40
```

В этом рабочем каталоге менеджер создал подкаталог www, а в нём — три текстовых файла index.md, content.md и style.md в кодировке UTF-8, содержащих информационные страницы, оформленные с помощью разметки Markdown.

Листинг файла index.md

```
Простой одностраничный веб-сайт
===============================

Простой одностраничный веб-сайт
===============================


Этот проект посвящён разработке шаблона простого одностраничного веб-сайта,
который может быть использован для быстрого создания сайта-визитки.

Ключевыми особенностями проекта являются:

 * следование веб-стандартам;
 * использование свободных средств разработки;
 * открытая лицензия.

Дополнительные сведения:

 * [Информационное наполнение]($ROOT/doc/$CURRENT/www/content.md)
 * [Стилевое оформление]($ROOT/doc/$CURRENT/www/style.md)

Обсуждение проекта ведётся [на форуме]($ROOT/forum).
```

Листинг файла content.md

```
Информационное наполнение
=========================

Информационное наполнение
=========================


Одностраничный веб-сайт состоит из заголовка, подвала и основного
содержимого, представленного информационными блоками. Каждый блок
содержит иллюстрацию и текст, которые чередуются в шахматном по-
рядке: у нечётных блоков иллюстрация располагается справа от тек-
ста, а у нечётных - слева.

В заголовке расположено меню, пункты которого являются гиперссыл-
ками на якоря, представленные заголовками информационных блоков.
При выборе пункта меню происходит прокручивание страницы к содер-
жимому выбранного информационного блока.
```

```
[Назад]($ROOT/home)
```

Листинг файла style.md

```
Стилевое оформление веб-странички
=================================

Стилевое оформление веб-странички
=================================

Правила оформления веб-странички находятся в файле _index.css_.

Чтобы веб-странички одинаково отображались в различных веб-
браузерах, первоначально производится обнуление значений полей и
отступов для всех элементов:

`
* {margin: 0; padding: 0}
`

Тексту документа назначается основной шрифт, размер которого при-
нимается за базовый. Размеры шрифтов остальных элементов веб-
странички задаются в относительных единицах, благодаря чему они
будут изменяться пропорционально изменению размеру базового шриф-
та.

`
body {font: normal normal normal 100% sans-serif}
`

Центрирование блочных элементов по горизонтали осуществляется с
помощью приёма, устанавливающего автоматические значения полей
слева и справа от центрируемого элемента:

`
.center {margin-left: auto; margin-right: auto}
`

[Назад]($ROOT/home)
```

Менеджер добавил созданные файлы в список системы контроля версий с помощью команды:

```
fossil add www
```

Команда сообщила об успешном выполнении:

```
ADDED www/content.md
ADDED www/index.md
ADDED www/style.md
```

После этого менеджер загрузил эти файлы в репозиторий командой:

```
fossil commit -m "Добавлены исходные варианты файлов сопроводи-
тельной документации"
```

Команда завершилась без ошибок:

```
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 0 received: 0
Pull done, sent: 520 received: 561 ip: 192.168.56.101
New_Version:
407ec7b25d56dbc8d4d4048b830621cc2f90dd7d8604244144e8d91bbd3c5aa6
Autosync: http://Manager@fossil.server:8081/
Round-trips: 1 Artifacts sent: 4 received: 0
Sync done, sent: 2555 received: 707 ip: 192.168.56.101
```

9.4.2. Использование документации

Теперь сопроводительная документация находится в репозитории. Чтобы убедиться в этом, достаточно в адресной строке веб-браузера набрать:

```
http://fossil.server:8081/doc/документация/www/index.md
```

Должна отобразиться страничка, показанная на рисунке 9.6.

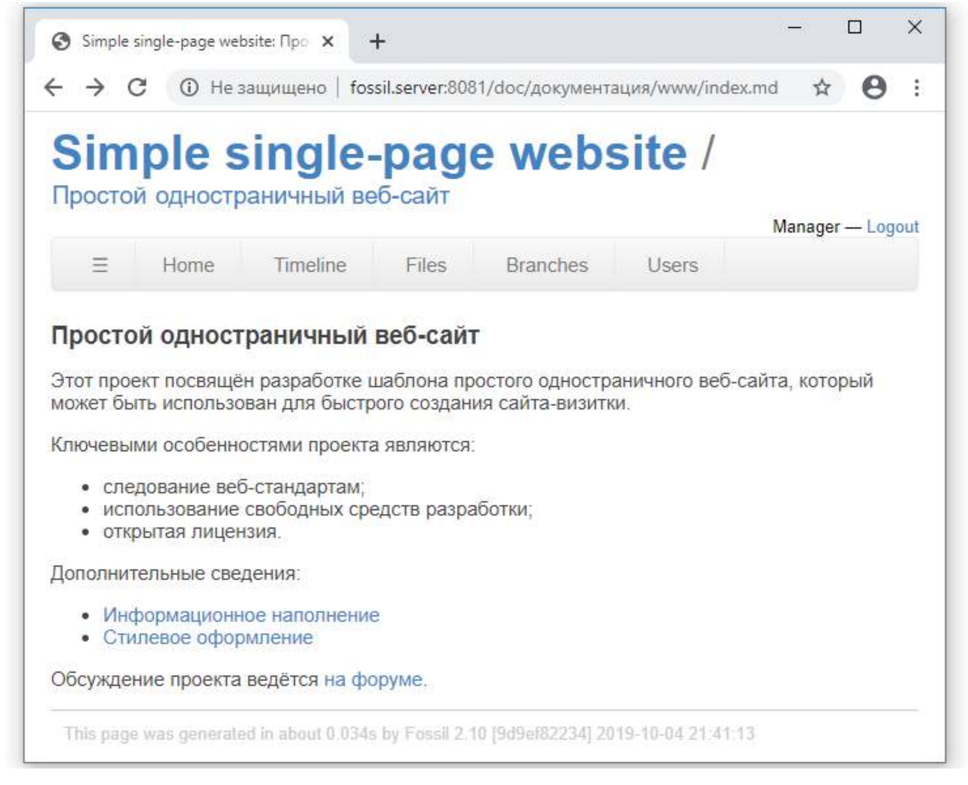


Рис. 9.6

Отображение страницы встроенной документации проекта

Разберём адресную строку подробно (рис. 9.7). Начало у неё традиционное: http://fossil.server:8081 — это протокол, сетевое имя сервера и номер TCP-порта, на котором ожидает сетевых подключений сервер Fossil. Средняя часть адреса /doc/документация задаёт ссылку на точку сохранения «документация» (как мы помним, имя ветви эквивалентно идентификатору самой поздней точки

сохранения на ней) и требование интерпретировать заключительную часть адреса в качестве пути в моментальном снимке рабочего каталога к файлу встроенной документации: /www/index.md.

http://fossil.server:8081/doc/документация/www/index.md



Рис. 9.7

Соответствие элементов адресной строки сущностям системы Fossil: веб-серверу, ветви репозитория и пути к файлу относительно моментального снимка

Веб-сервер Fossil обрабатывает эту адресную строку и возвращает браузеру веб-страницу, сгенерированную с использованием содержимого файла index.md. При этой обработке первый заголовок из файла документации используется в качестве подзаголовка репозитория и отображается над строкой главного меню. Поэтому в файле документации первые заголовки продублированы.

В файлах документации имеются гиперссылки на внутренние элементы репозитория — на другие файлы документации и на стартовую страницу. Для универсальности в этих ссылках используются переменные $ROOT и $CURRENT. При подготовке веб-страниц системой Fossil эти переменные заменяются на сетевой адрес репозитория и на имя текущей ветви репозитория соответственно.

Чтобы установить страницу встроенной документации в качестве стартовой страницы репозитория, надо завершить процедуру его настройки, начатую ещё в процессе его создания. Для этого придётся авторизоваться в системе под именем пользователя — владельца SetupUser (прав администратора репозитория недостаточно) и воспользоваться пунктом меню Admin | Configuration (рис. 9.8).

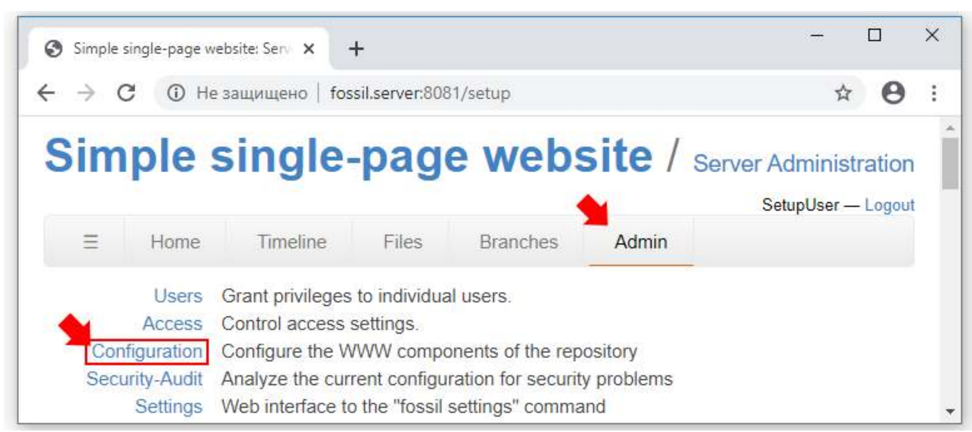


Рис. 9.8

Переход к настройке репозитория

На открывшейся форме надо внести адрес требуемой страницы документации в поле Index Page (рис. 9.9). После этого назначенная страница будет показываться при запуске веб-интерфейса репозитория и при выборе пункта меню Home.

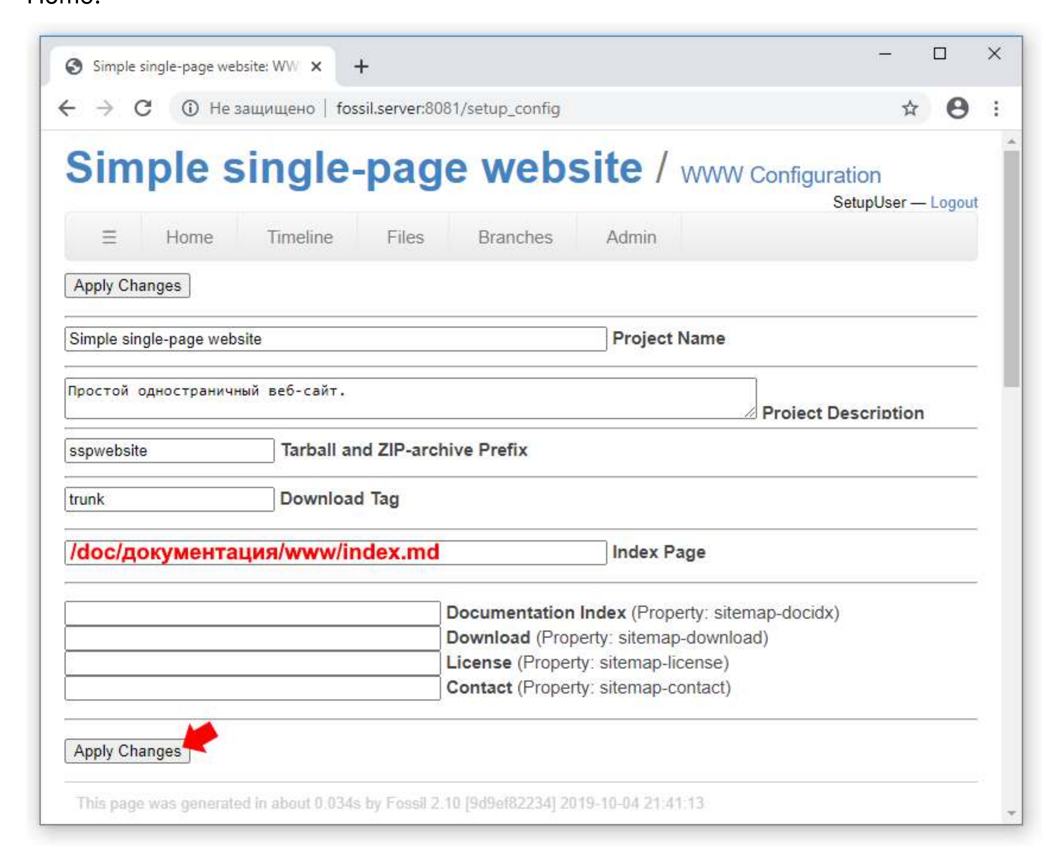


Рис. 9.9

Указание адреса встроенной страницы документации для использования в качестве стартовой страницы репозитория

9.5. Темы оформления

Система Fossil позволяет в широких пределах настраивать оформление веб-интерфейса, изменяя цветовые схемы, шрифты, местоположение и состав интерфейсных элементов. Всё это реализуется с помощью тем — совокупностей шаблонов веб-страниц и таблиц стилей. В поставку Fossil помимо основной темы входят ещё двенадцать предварительно настроенных тем.

9.5.1. Настройка темы

Для переключения темы оформления надо в адресной строке веб-браузера набрать: http://fossil.server:8081/setup_skin_admin. Откроется страничка со списком предустановленных тем (рис. 9.10). Текущая тема обозначена надписью

Currently In Use (Используется сейчас). Активация требуемой темы производится нажатием кнопки Install, расположенной справа от её наименования.

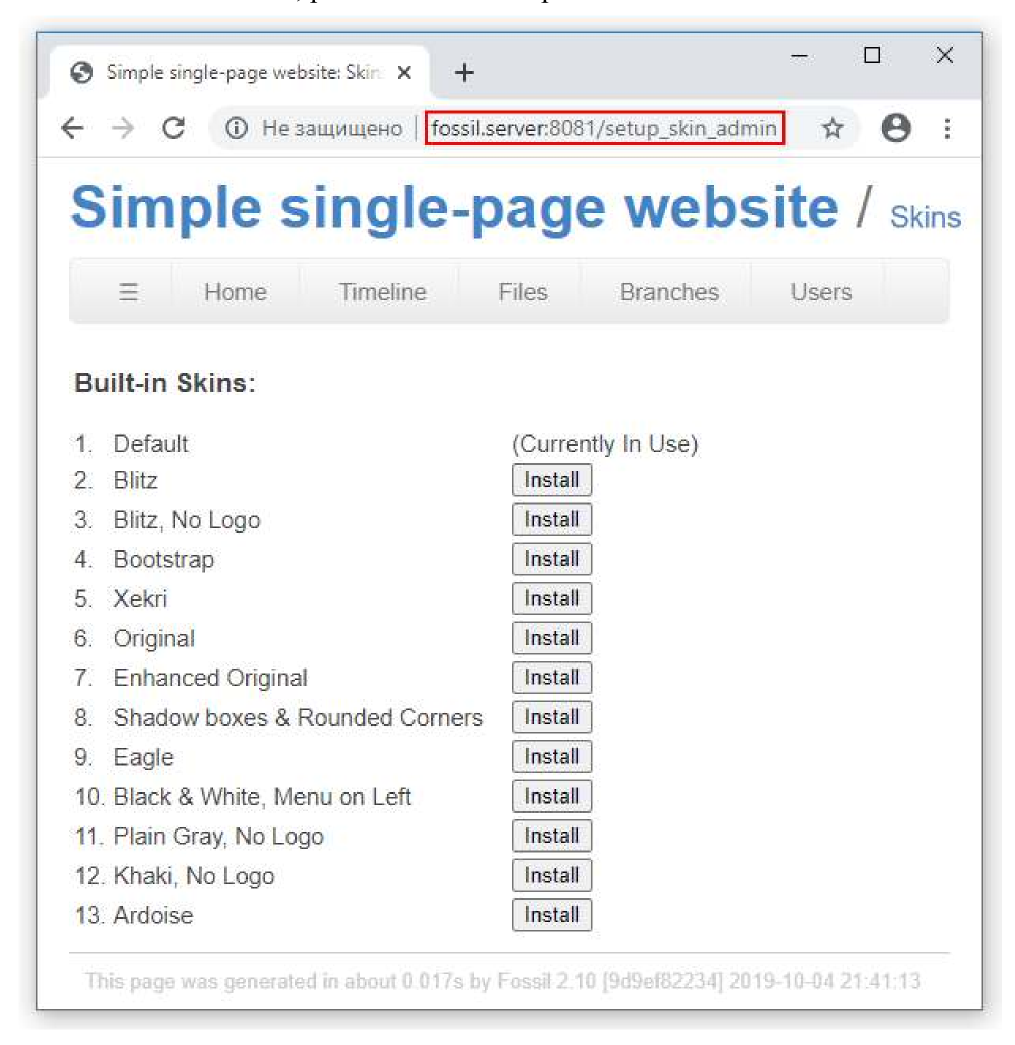


Рис. 9.10

Страничка управления темами оформления веб-интерфейса Fossil

Например, в результате активации темы Khaki, No Logo веб-интерфейс приобретает вид, показанный на рисунке 9.11. Если по какой-либо причине выбранная тема не приглянулась (например, в теме Khaki из главного меню исчез пункт вызова раскрывающегося меню), то для возврата к исходной теме надо перейти на страничку со списком тем, как было описано выше, и активировать первую тему Default.

Дополнительную гибкость интерфейсу Fossil придаёт возможность редактирования шаблонов веб-страниц и их стилей. Для перехода к настройке интерфейса надо выбрать в раскрывающемся меню пункт Administration pages, после чего перейти по гиперссылке Skins (рис. 9.12).

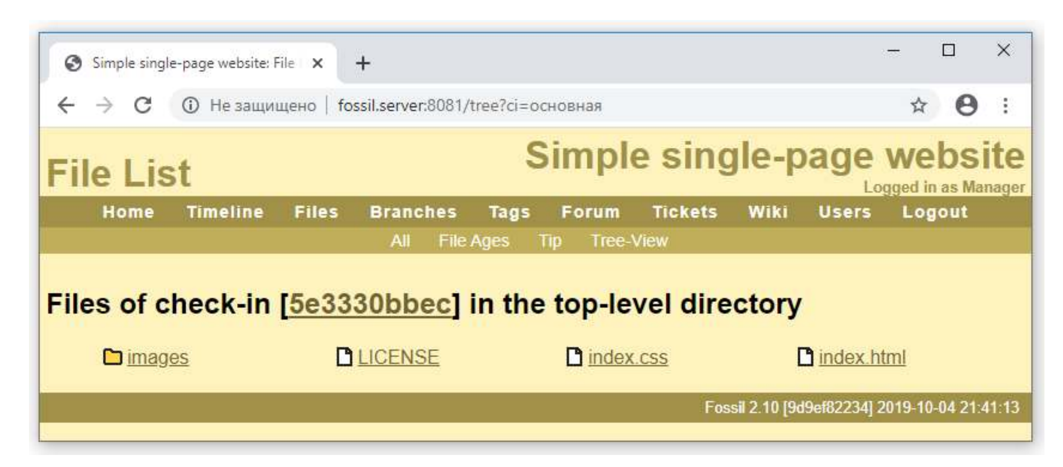


Рис. 9.11

Веб-интерфейс с активной темой Khaki

Simple single-page website: Serv
Не защищено | fossil.server:8081/setup
Simple single-page website / Server Administration
Manager — Logout
Home Timeline Files Branches Users
Home Page
File Browser
Tree-view, Trunk Check-in
Flat-view
File ages for Trunk
Unversioned Files
Forum
Recent activity
Tickets
Logout
Change Password
Repository Status
Collisions on hash prefixes
List of URLs used to access this repository
Administration Pages
Pending Moderation Requests
Admin log
Users Grant privileges to individual users.
Security-Audit Analyze the current configuration for security problems
Timeline Timeline display preferences
Search Configure the built-in search engine
URL Aliases Configure URL aliases
Skins Select and/or modify the web interface "skins"
Moderation Enable/Disable requiring moderator approval of Wiki and/or Ticket changes and attachments.

Рис. 9.12

Переход к настройке интерфейса

Откроется страница, на которой процедура редактирования темы оформления представлена в виде восьми шагов. Вносить изменения прямо в активную тему нельзя. Поэтому на первом шаге предлагается выбрать один из девяти сло-

тов-черновиков, в которых будут сохраняться выполненные изменения (рис. 9.13).

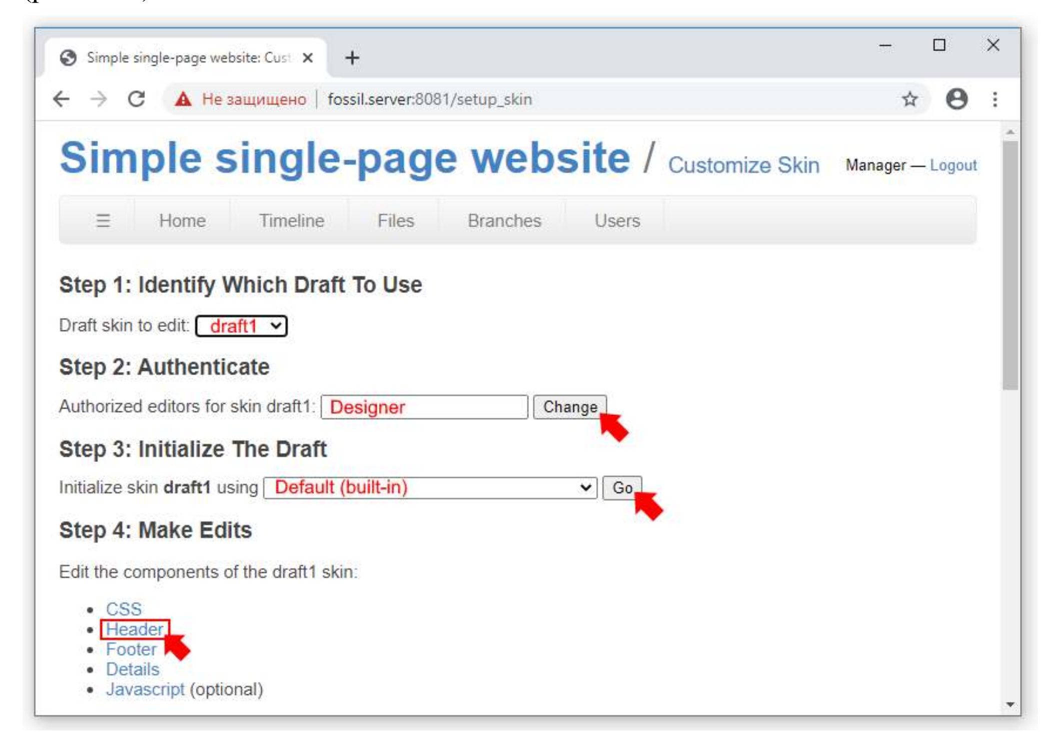


Рис. 9.13

Первые четыре шага настройки темы оформления

Если настройкой занимается администратор системы, то на втором шаге предоставляется возможность перечислить через запятую имена учётных записей пользователей системы Fossil, которые будут иметь доступ к редактированию выбранного черновика. Ввод пользователей надо завершить нажатием кнопки Change.

Если работает авторизованный администратором пользователь, то ему сообщается о наличии разрешения на редактирование и предлагается перейти к следующему шагу. Если же текущий пользователь не авторизован, то ему рекомендуется обратиться к администратору.

На третьем шаге можно выбрать одну из предустановленных тем или другой черновик для использования в качестве основы создаваемой темы. Вариант Currently In Use (Используется сейчас) соответствует активной теме веб-интерфейса. Смену темы надо подтвердить нажатием кнопки Go.

На четвёртом шаге предлагается заняться собственно редактированием стилей и шаблонов страниц. Для перехода в режим редактирования можно использовать одну из гиперссылок:

- CSS — редактирование таблицы стилей;
- Header — редактирование заголовка страниц;
- Footer — редактирование подвала страниц;

- Details — редактирование настроек;
- Javascript — редактирование сценариев JavaScript.

Поскольку этот пункт является ключевым во всём процессе, рассмотрим его подробнее.

9.5.2. Редактирование шаблонов

Шаблоны веб-страниц Fossil представляют собой HTML-код с встроенными сценариями на языке TH1, который является минималистичной реализацией языка сценариев Tcl. Текст сценариев заключается в теги <TH1> ... </TH1>.

Чтобы изменить главное меню веб-интерфейса, надо подправить шаблон заголовка страниц, для чего следует воспользоваться гиперссылкой Header. Откроется страница с редактором кода заголовка (рис. 9.14). На этой странице под главным меню имеется дополнительное меню, которое позволяет переключаться между редактируемыми элементами шаблона, что избавляет от необходимости возвращаться в окно настроек шаблона со списком гиперссылок.

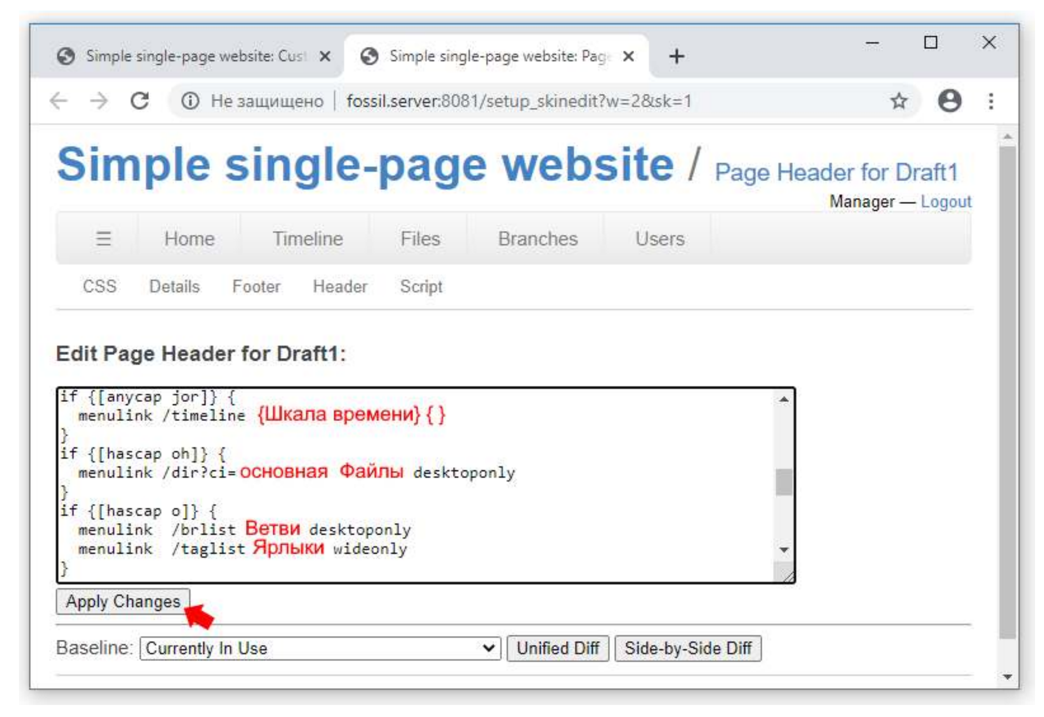


Рис. 9.14

Страница редактирования шаблона заголовка

Поскольку состав меню зависит от прав пользователя, работающего с веб-интерфейсом, оно реализовано в виде сценария. Элементы меню описываются условными конструкциями вида:

```
if {[hascap РАЗРЕШЕНИЯ]} {
  menulink АДРЕС ТЕКСТ СТИЛЬ
}
…
if {[anycap РАЗРЕШЕНИЯ]} {
  menulink АДРЕС ТЕКСТ СТИЛЬ
}
```

Операторы if проверяют наличие у текущего пользователя необходимых для доступа к пункту меню разрешений. Выражение hascap РАЗРЕШЕНИЯ является истинным, если у пользователя имеются все разрешения, представленные символами, входящими в строку РАЗРЕШЕНИЯ. Выражение anycap РАЗРЕШЕНИЯ является истинным, если у пользователя есть хотя бы одно разрешение, представленное входящим в строку РАЗРЕШЕНИЯ символом. Пункт меню, задаваемый оператором menulink, добавляется в меню только в том случае, если соответствующее ему проверяемое условие истинно.

В операторе menulink АДРЕС ТЕКСТ СТИЛЬ параметр АДРЕС — это адрес страницы веб-интерфейса, соответствующей пункту меню, ТЕКСТ — это текст, который будет отображаться в соответствующей позиции меню, а СТИЛЬ — это CSS-стиль, который применяется к пункту меню. Стили wideonly и desktoponly определяются в медиазапросах CSS, проверяющих ширину окна веб-браузера, и при недостаточной её величине скрывают своё содержимое. Можно указать пустое значение стиля в виде { }.

Можно, например, русифицировать главное меню, заменив текст его пунктов следующим образом: Home — Старт, Timeline — Шкала времени, Files — Файлы, Branches — Ветви, Tags — Ярлыки, Forum — Форум, Tickets — Заявки, Wiki — Документы, Admin — Настройка, Users — Пользователи. Поскольку текст Шкала времени состоит из двух слов, в операторе menulink его надо заключить в фигурные скобки:

```
menulink /timeline {Шкала времени} {}
```

Чтобы файловый обозреватель всегда открывался на основной ветви проекта, а не на той, в которой позже всех была создана точка сохранения, в качестве адреса для пункта меню Файлы вместо /dir?ci=tip надо указать /dir?ci=основная:

```
menulink /dir?ci=основная Файлы desktoponly
```

После корректировки шаблона надо нажать на страничке редактирования заголовка кнопку Apply Changes. Можно сравнить исправленный шаблон с исходным или шаблоном из другой темы. Для этого надо в поле Baseline (Основа) выбрать интересующую тему и нажать кнопку Unified Diff. Внизу страницы отобразятся имеющиеся различия (рис. 9.15).

Рис. 9.15

Просмотр различий между изменённым и исходным шаблонами

После фиксации изменений в черновике рядом с кнопкой Apply Changes появляется кнопка Revert To "current", которая позволяет вернуть редактируемый шаблон к исходному виду.

9.5.3. Использование темы

Продолжим рассмотрение процедуры настройки веб-интерфейса (рис. 9.16). На пятом шаге после внесения изменений в шаблоны и стили предлагается оценить получившийся результат, воспользовавшись одной из гиперссылок. При этом веб-страничка, соответствующая использованной гиперссылке, открывается в отдельной вкладке браузера.

Если результат неудовлетворительный, то шестым шагом предписывается повторить правку черновика темы, вернувшись к четвёртому и пятому шагам.

На седьмом шаге предлагается зафиксировать внесённые изменения в черновике темы и опубликовать получившийся результат. Для этого на страничке настройки интерфейса имеются флажки:

- Skin draft1 has been tested and found ready to production — подтверждение готовности черновика темы к эксплуатации;
- The current skin should be overwritten with draft1 — требование заменить текущую тему темой из черновика.

Выполнение отмеченных действий происходит по нажатию на кнопку Publish Draft1 (Опубликовать Черновик1). В результате произведенных изменений главное меню приобрело вид, показанный на рисунке 9.17.

Simple single-page website: Cust

Не защищено | fossil.server:8081/setup_skin

Step 5: Verify The Draft Skin

To test this draft skin, insert text "/draft1/" just before the operation name in the URL. Here are a few links to try:

- http://fossil.server:8081/draft1/home
- http://fossil.server:8081/draft1/timeline
- http://fossil.server:8081/draft1/dir?ci=tip
- http://fossil.server:8081/draft1/dir?ci=tip&type=tree
- http://fossil.server:8081/draft1/brlist
- http://fossil.server:8081/draft1/info/trunk

You will probably need to press Reload on your browser before any CSS changes will take effect.

Step 6: Iterate

Repeat step 4 and step 5 as many times as necessary to create a production-ready skin.

Step 7: Publish

When the draft1 skin is ready for production use, make it the default scan by clicking the acknowledgements and pressing the button below:

Skin draft1 has been tested and found ready for production.
The current skin should be overwritten with draft1.
Publish Draft1

You will probably need to press Reload on your browser after publishing the new skin.

Step 8: Cleanup and Undo Actions

Visit the Skin Admin page for cleanup and recovery actions.

This page was generated in about 0.034s by Fossil 2.10 [9d9ef82234] 2019-10-04 21:41:13

Рис. 9.16

Продолжение настройки веб-интерфейса

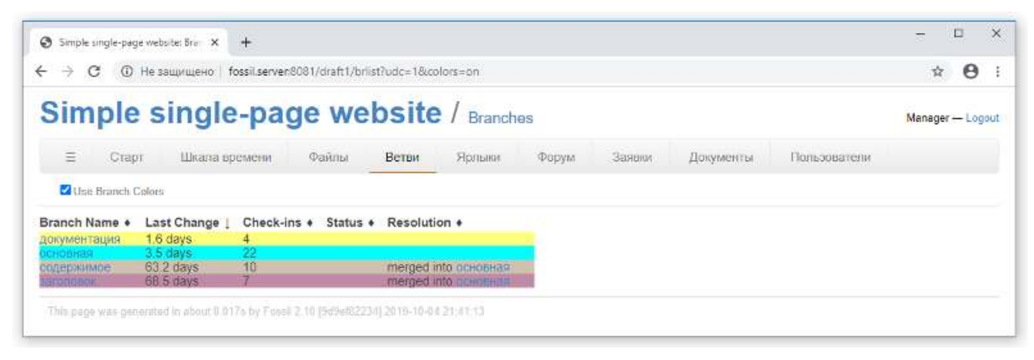


Рис. 9.17

Русифицированное главное меню веб-интерфейса

На восьмом шаге приводится гиперссылка Skin Admin, ведущая на страничку управления темами, которая упоминалась в самом начале этого раздела. Теперь на ней появились дополнительные элементы управления (рис. 9.18).

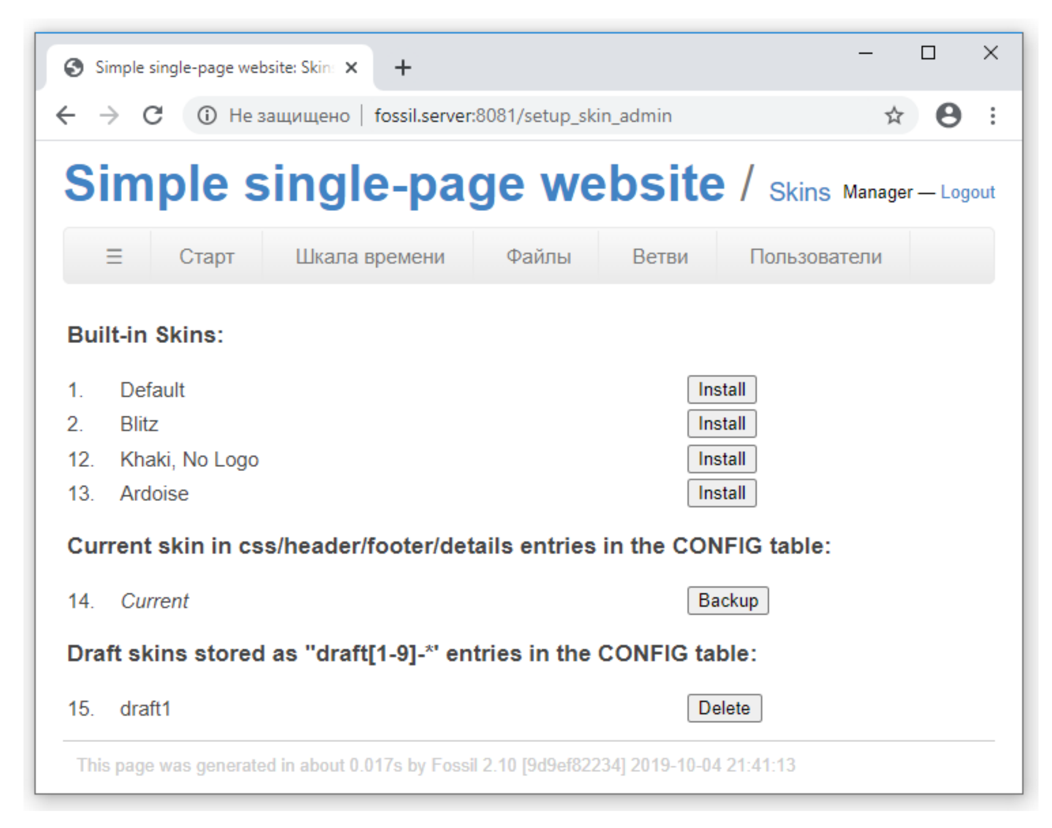


Рис. 9.18

Обновлённая веб-страничка управления темами

В блоке, озаглавленном Current skin in css/header/footer/details entries in the CONFIG table, находится запись о текущей теме. Рядом с ней имеется кнопка Backup, позволяющая сохранить эту тему под своим оригинальным именем. При нажатии на эту кнопку запрашивается имя для текущей темы, которое присваивается ей после нажатия на кнопку Save (рис. 9.19). В дальнейшем этот вариант темы оформления можно переименовать с помощью кнопки Rename.

Внизу веб-странички управления темой имеется блок Draft skins stored as "draft[1-9]-*" entries in the CONFIG table, в котором перечислены модифицированные черновики тем. Рядом с именем черновика имеется кнопка Delete, позволяющая его удалить, если дальнейшая доработка содержащейся в нём темы оформления не требуется. При этом изготовленная на основе удалённого черновика тема сохранится и может продолжать использоваться.

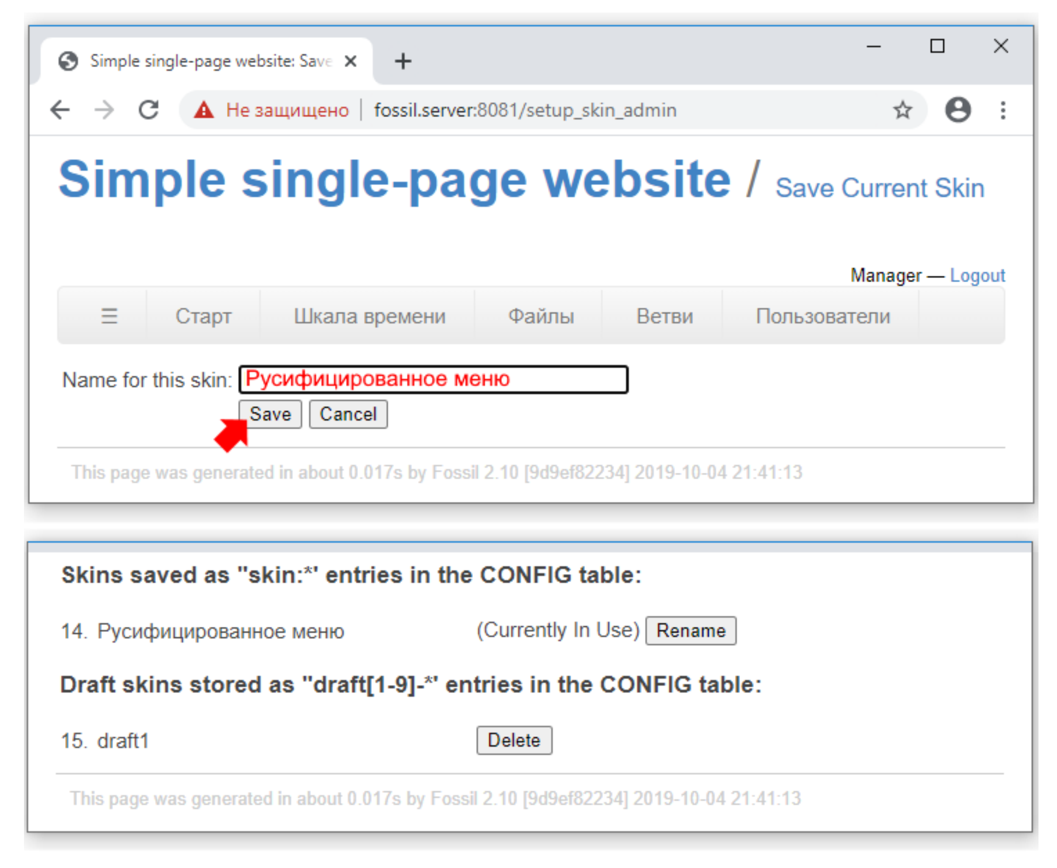


Рис. 9.19

Присвоение оригинального имени созданной теме

9.6. Механизм поиска

Полезность информационной системы во многом определяется возможностью быстрого отыскания требуемых сведений в массиве хранящихся в ней данных. С течением времени репозиторий становится ценнейшей коллекцией разнообразных сведений о проекте, содержащихся не только в рабочих файлах, но и в страницах документации, заявках на доработку и в обсуждениях на форуме. Только отыскать нужный текст бывает непросто. Перекрёстные гиперссылки между подсистемами сильно помогают в этом вопросе, но не всегда удаётся найти тот конец нити, по которому можно распутать клубок.

В системе Fossil имеется возможность задействовать механизм полнотекстового поиска, реализованный в базе данных SQLite, по информации, накопленной в её подсистемах. Чтобы это осуществить, надо воспользоваться пунктом раскрывающегося меню Administration Pages, после чего перейти по гиперссылке Search (рис. 9.20).

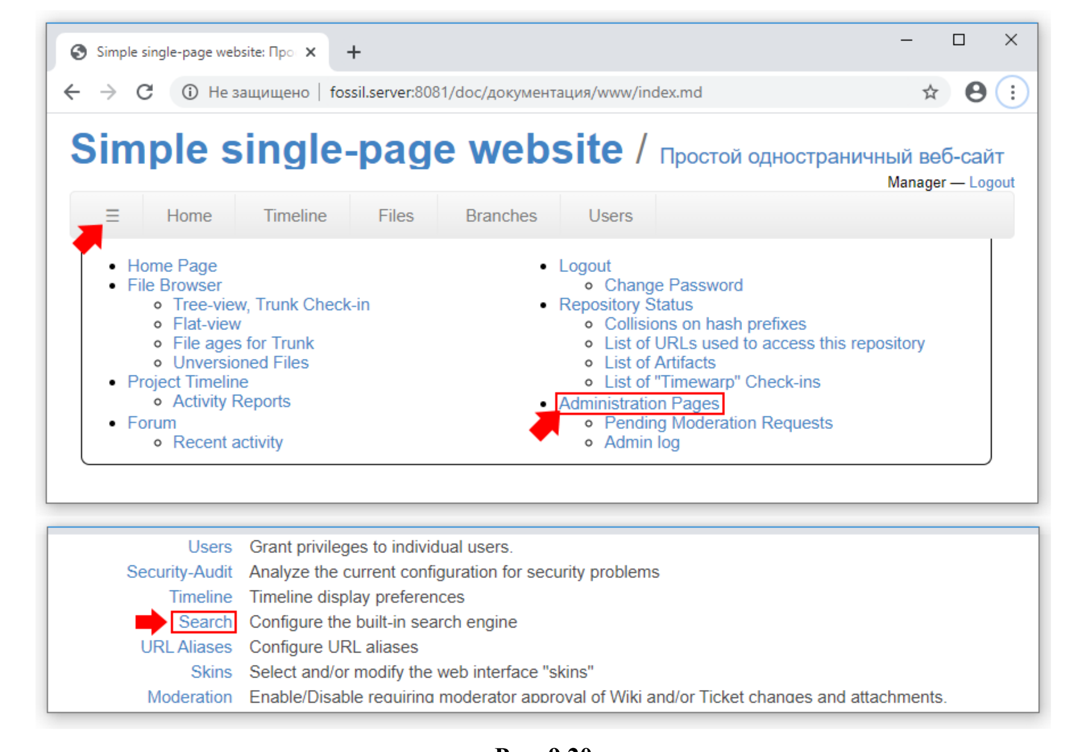


Рис. 9.20

Переход к настройке поисковой системы

Откроется страница настройки поисковой системы Fossil (рис. 9.21). Первые два поля предназначены для настройки поиска внутри документов — файлов, хранящихся в репозитории. В поле Document Glob List надо указать маски имён файлов, в содержимом которых будет осуществляться поиск. В поле Document Branch требуется указать имя ветви репозитория, файлы которой будут участвовать в поиске. Если хотя бы одно из этих полей оставить пустым, то поиск по документам производиться не будет.

Далее следует группа флажков, с помощью которых можно отметить информационные хранилища подсистем Fossil, участвующие в поиске:

- Search Check-in Comments — комментарии к точкам сохранения в репозитории;
- Search Documents — документы (файлы, о которых говорилось выше);
- Search Tickets — заявки на доработку;
- Search Wiki — статьи системы документирования;
- Search Tech Notes — технические заметки;
- Search Forum — обсуждения на форуме.

Чтобы выполненные настройки вступили в силу, надо нажать кнопку Apply Changes. С этого момента на страницах веб-интерфейса выбранных подсистем появится строка поиска, в которой можно набирать искомые слова (рис. 9.22).

Simple single-page website: Sear ×

Не защищено | fossil.server:8081/srchsetup

Simple single-page website / Search Configuration

Manager — Logout

☰ Home Timeline Files Branches Users

Server-specific settings that affect the /search webpage.

*.md, *.html

Document Glob List

The "Document Glob List" is a comma- or newline-separated list of GLOB expressions that identify all documents within the source tree that are to be searched when "Document Search" is enabled. Some examples:

.wiki,.html,*.md,*.txt	Search all wiki, HTML, Markdown, and Text files
doc/*.md,*/README.txt,README.txt	Search all Markdown files in the doc/ subfolder and all README.txt files.
*	Search all checked-in files
(blank)	Search nothing. (Disables document search).

документация **Document Branch**

When searching documents, use the versions of the files found at the type of the "Document Branch" branch. Recommended value: "trunk". Document search is disabled if blank.

- [x] **Search Check-in Comments**
- [x] **Search Documents**
- [x] **Search Tickets**
- [x] **Search Wiki**
- [x] **Search Tech Notes**
- [x] **Search Forum**

Apply Changes

The SQLite FTS4 search index is disabled. All searching will be a full-text scan. This usually works fine, but can be slow for larger repositories.

- [ ] **Use Porter Stemmer**

Create A Full-Text Index

This page was generated in about 0.017s by Fossil 2.10 [9d9ef82234] 2019-10-04 21:41:13

Рис. 9.21

Страница настройки поисковой системы

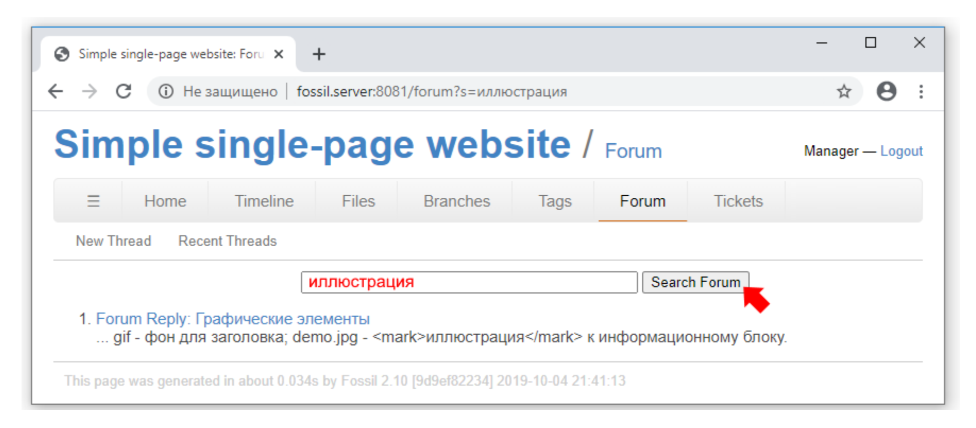


Рис. 9.22

Строка поиска на странице форума

Помимо этого начинает работать специальная страница поиска, доступная по адресу http://fossil.server:8081/search/ (рис. 9.23). С этой страницы можно запустить поиск сразу по всем подсистемам или выбрать интересующую область поиска из списка.

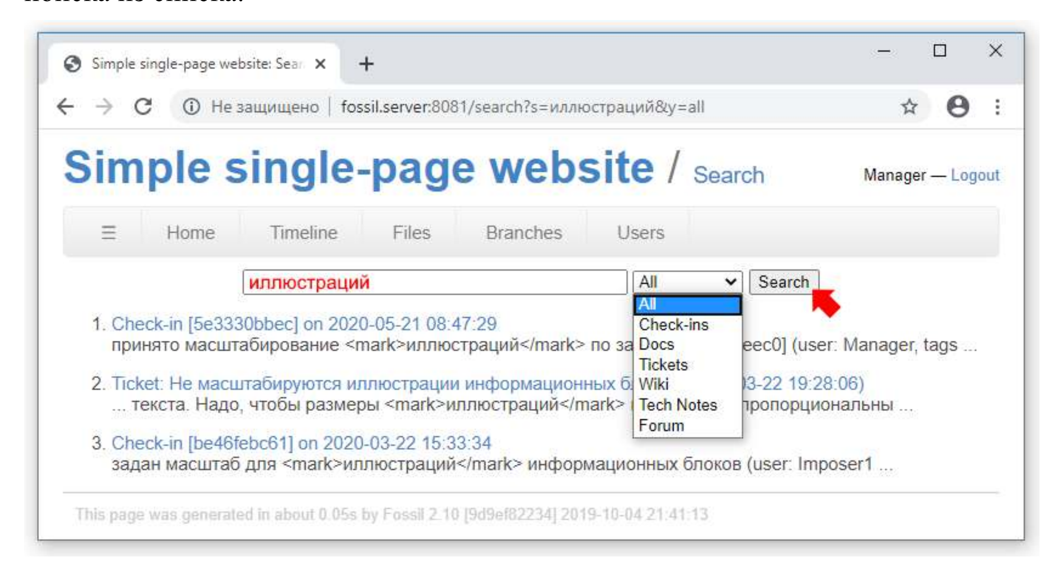


Рис. 9.23

На странице поиска можно искать информацию сразу во всех подсистемах Fossil

С произведенными выше настройками поиск выполняется путём последовательного просмотра текстов, хранящихся в выбранных подсистемах, и сравнения их фрагментов со словом, введённым в строке поиска. Такой метод может занимать длительное время на больших репозиториях. Для увеличения эффективности поиска можно создать индекс — словарь, в котором в определённом порядке представлены ссылки на сведения, имеющиеся в информационных хранилищах. Использование индекса существенно ускоряет процесс, но для хранения словаря используется дополнительное пространство на носителе информации.

При создании индекса можно задействовать алгоритм, позволяющий обнаруживать в тексте различные языковые формы искомого слова, что расширяет поисковую выдачу. К сожалению, имеющаяся в Fossil реализация этого алгоритма не работает для русского языка.

Чтобы создать поисковый индекс и задействовать его в поиске, надо на странице настроек нажать кнопку Create A Full-Text Index. Если при этом будет установлен флажок Use Porter Stemmer, то при создании индекса будет использован алгоритм Портера.

9.7. Объявления

Иногда возникает необходимость уведомить всех участников проекта о каком-нибудь мероприятии. Например, о проведении конференции по вопросу, касающемуся разработки проекта, о необходимости подачи сведений для со-

ставления графика отпусков, об изменении режима допуска в офисное здание или о предстоящем ремонте автомобильной парковки. Вариантов донесения информации до сотрудников известно множество: можно организовать собрание руководителей подразделений и поручить им проинформировать своих подчинённых, можно обойти или обзвонить все кабинеты, можно уведомлять всех на входе в здание, а можно вывесить объявление.

Вариант с объявлением выглядит наиболее привлекательным и наименее затратным. Объявление можно перечитать несколько раз, к нему можно вернуться по прошествии времени для уточнения забытой информации. Хорошо, если в офисном помещении есть специально организованная доска объявлений, с содержимым которой все привыкли знакомиться. А если нет?

Система Fossil позволяет разместить короткое объявление прямо на страницах своего веб-интерфейса. Если к этому времени Fossil стал привычным рабочим инструментом, то такой способ оповещения вряд ли останется незамеченным.

Чтобы разместить объявление, надо воспользоваться пунктом раскрывающегося меню Administration Pages, после чего перейти по гиперссылке Ad-Unit (рис. 9.24).

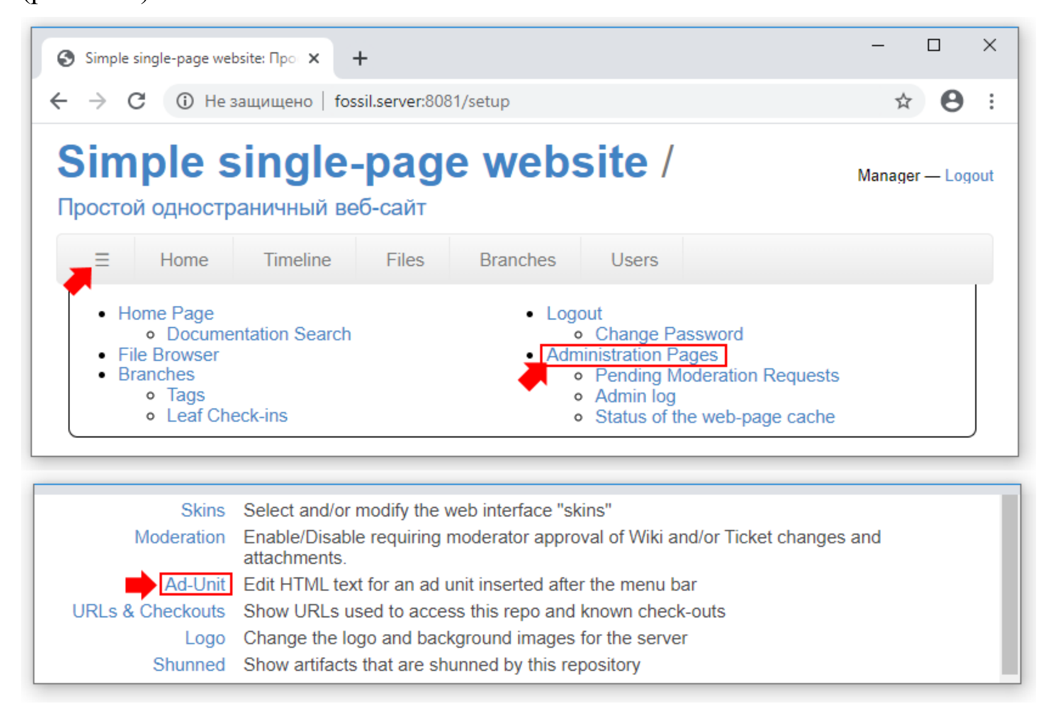


Рис. 9.24

Переход к размещению объявления

Откроется форма размещения объявления (рис. 9.25). На ней имеются два больших текстовых поля. Первое поле, озаглавленное Banner Ad-Unit, предназначено для ввода HTML-кода узкого горизонтального баннера, который будет размещаться под горизонтальным меню. Второе поле, озаглавленное Right-Column

Ad-Unit, предназначено для ввода HTML-кода вертикального блока, который будет размещаться сбоку от основного содержимого. На одной странице будет отображено только одно объявление — горизонтальное или вертикальное. Вертикальный вариант используется на страницах с узким содержимым — Files (Файлы) и Tags (Ярлыки). Чтобы активировать внесённые изменения, надо нажать кнопку Apply Changes.

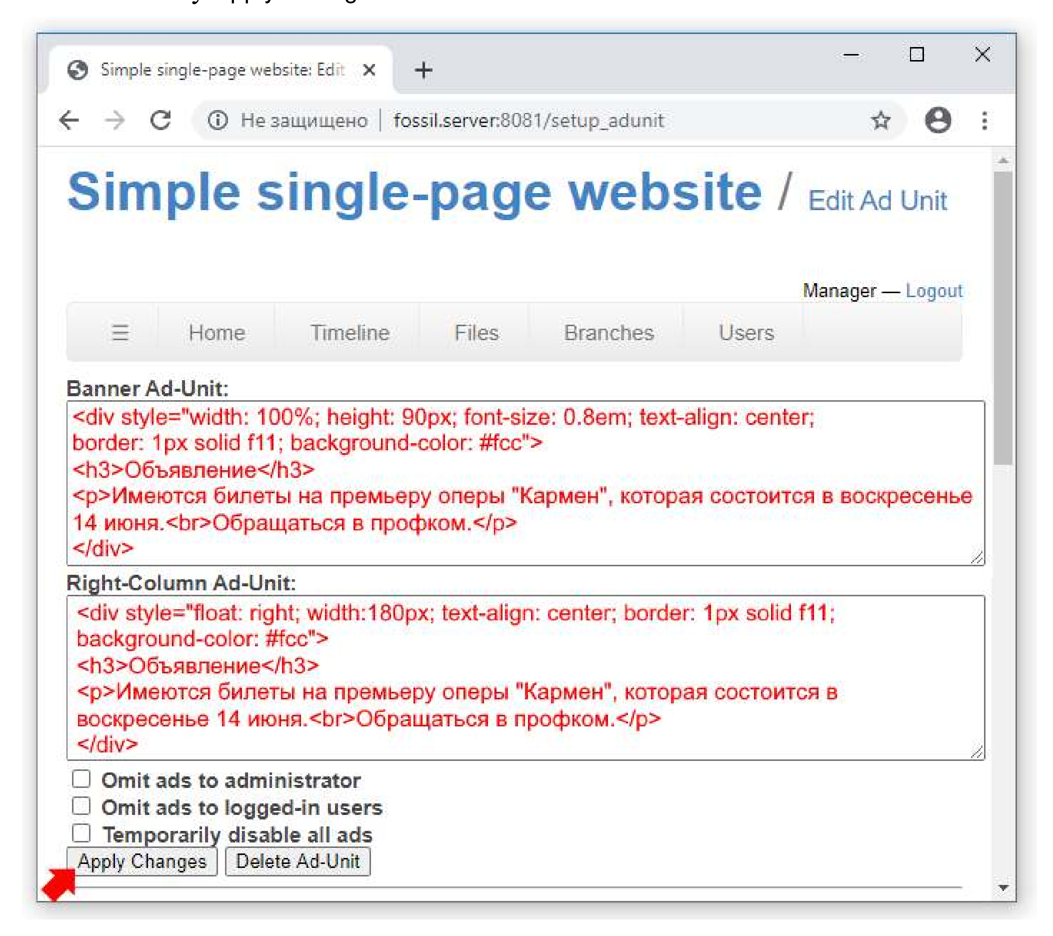


Рис. 9.25

Форма размещения объявления

В нижней части формы (которая не показана на рисунке) имеются рекомендации относительно оформления объявления, в частности рекомендуется дописать CSS-стили для некоторых классов в редакторе темы веб-интерфейса. Но можно обойтись стилями, встроенными в теги HTML. Например, горизонтальный баннер может быть оформлен так:

```
<div style="width: 100%; height: 90px; font-size: 0.8em; text-align: center;
border: 1px solid f11; background-color: #fcc">
<h3>Объявление</h3>
<p>Имеются билеты на премьеру оперы "Кармен", которая состоится в
```

```
воскресенье 14 июня.<br>Обращаться в профком.</p>
</div>
```

А код вертикального блока может быть таким:

```
<div style="float: right; width:180px; text-align: center;
border: 1px solid f11; background-color: #fcc">
<h3>Объявление</h3>
<p>Имеются билеты на премьеру оперы "Кармен", которая состоится в
воскресенье 14 июня.<br>Обращаться в профком.</p>
</div>
```

Тогда объявление на страницах веб-интерфейса Fossil будет выглядеть, как показано на рисунке 9.26.

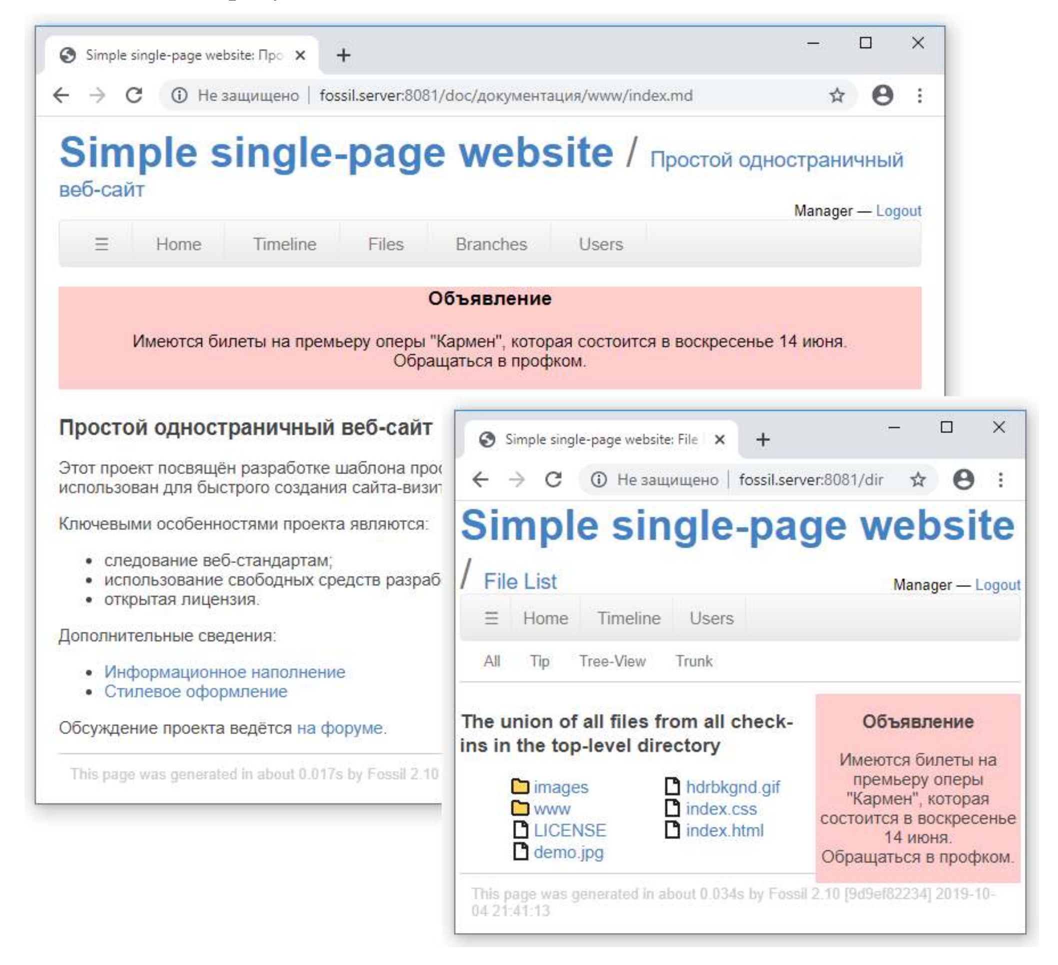


Рис. 9.26

Горизонтальный баннер (слева вверху) и вертикальный блок (справа внизу) с объявлением

На форме размещения объявления имеются флажки, позволяющие отключить показ объявлений:

- Omit ads to administrator — для администраторов системы Fossil;

- Omit ads to logged-in users — для всех пользователей, которые авторизовались в Fossil;
- Temporarily disable all ads — для всех посетителей веб-интерфейса.

Последний флажок удобно использовать, если объявление надо подготовить заранее, а демонстрировать пользователям — позже.

9.8. Обозреватель файлов

Рабочие файлы проекта, соответствующие любому историческому этапу разработки, могут быть выгружены из репозитория в рабочий каталог с помощью команды fossil checkout. Для участников проекта такой способ не составит труда, но стороннему наблюдателю перед этим придётся разобраться с системой Fossil и настроить индивидуальное рабочее место. Если у кого-то возникла необходимость в однократном получении какого-нибудь файла из репозитория, описанный способ требует слишком больших усилий.

К счастью, сервер Fossil предоставляет возможность доступа к файлам, хранящимся в репозитории, через веб-интерфейс. Для доступа к файловому архиву служит пункт главного меню Files или пункт раскрывающегося меню File Browser (рис. 9.27).

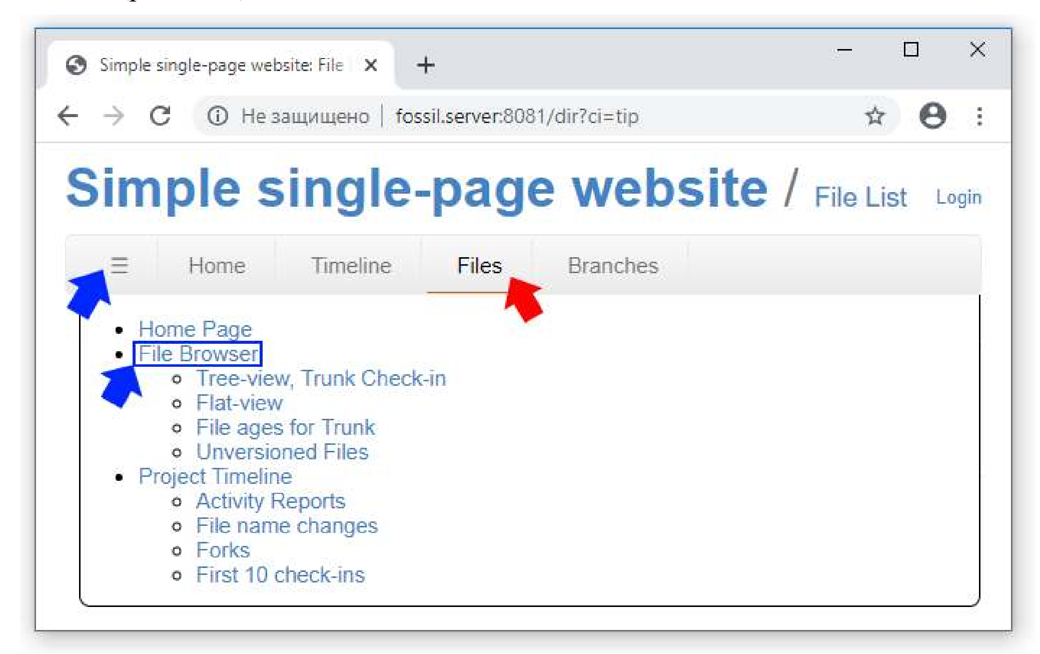


Рис. 9.27

Переход на страницу файлового архива

9.8.1. Просмотр списков файлов

На открывшейся после выбора пункта меню веб-странице будет отображён список файлов и каталогов, соответствующих самой поздней точке сохранения, созданной в репозитории. Если эта точка сохранения находится не на ос-

новной ветви разработки, то не стоит удивляться тому, что в этом списке отсутствуют все файлы рабочего каталога проекта. В этом случае можно воспользоваться пунктом дополнительного меню All, чтобы система Fossil объединила в одном списке самые последние версии файлов из всех точек сохранения (рис. 9.28).

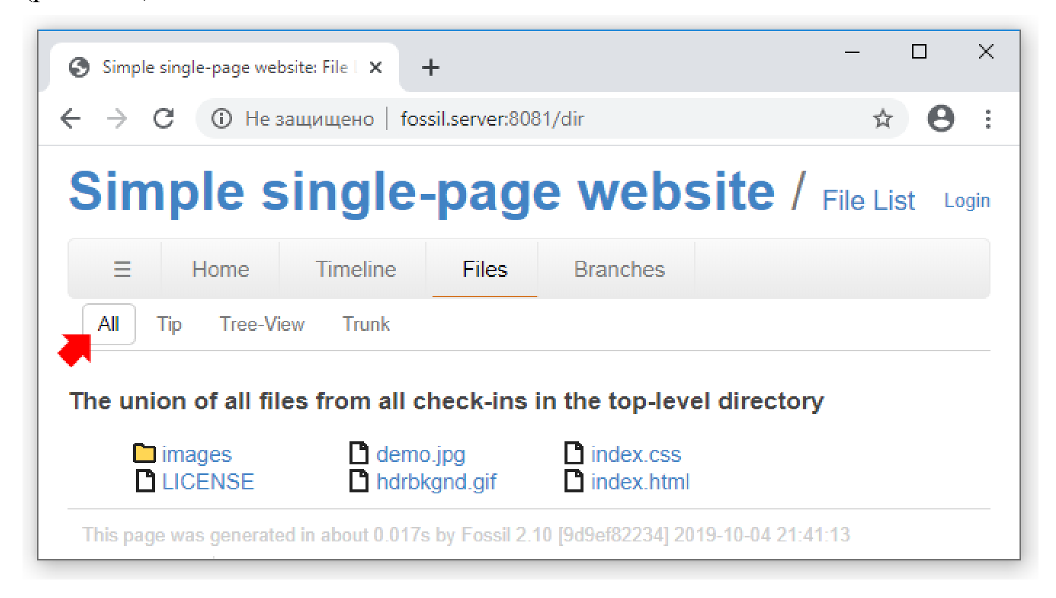


Рис. 9.28

Для отображения последних версий всех имеющихся в репозитории файлов можно воспользоваться пунктом дополнительного меню All

Неудобство этого режима состоит в том, что в каталогах отображаются даже перемещённые или удалённые из них файлы. Например, в корне проекта присутствуют файлы demo.jpg и hdrbkgnd.gif, которые при реструктуризации были перемещены в подкаталог images. Конечно, последние варианты этих файлов имеются в подкаталоге images, но такой вариант отображения может затруднить поиск нужного файла по его имени.

Зато в этом режиме в дополнительном меню появляется пункт Trunk, с помощью которого можно посмотреть файлы, соответствующие последней точке сохранения основной ветви разработки. Обычно это именно то, что требуется.

Если основная ветвь была переименована, то этот пункт меню не сработает (что ещё раз говорит в пользу сохранения имени основной ветви). Тогда остаётся только явно указать наименование ветви в адресной строке веб-браузера: http://fossil.server:8081/dir?ci=основная, где основная — это название интересующей ветви. Вместо названия ветви можно указать идентификатор любой точки сохранения, и тогда будет открыт список относящихся к ней файлов.

Вернуться в режим отображения файлов, соответствующих последней точке сохранения, можно с помощью пункта дополнительного меню Tip.

Файловый менеджер системы Fossil позволяет переключать режимы отображения. Так, с помощью пункта меню Tree-View можно включить режим отображения файловой иерархии (рис. 9.29) с возможностью сортировки по имени файлов или по времени их модификации, используя выбор из списка с вариантами Sort By Filename и Sort By Time.

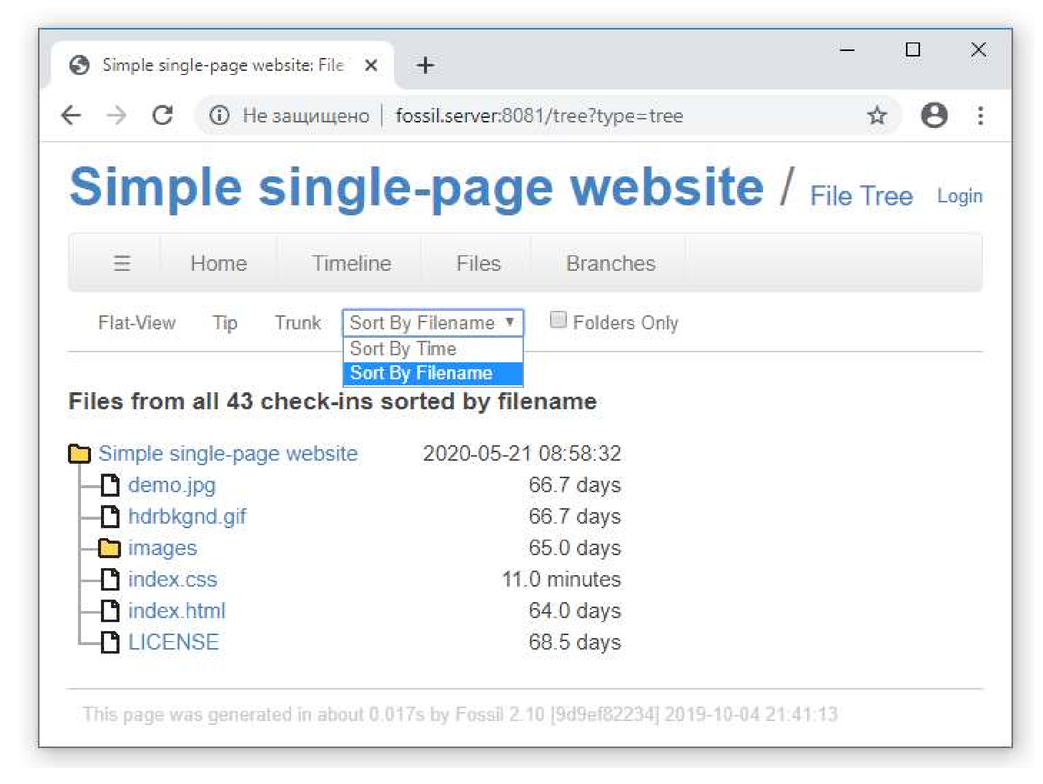


Рис. 9.29

Иерархический (древовидный) режим отображения файлов, который включается с помощью пункта меню Tree View. Можно упорядочивать файлы по имени (Sort By Filename) или по времени модификации (Sort By Time). Скрыть простые файлы и оставить только каталоги позволяет флажок Folders Only

Для упрощения навигации в больших проектах можно установить флажок Folders Only, в результате чего в списке останутся только каталоги. После перехода в нужный подкаталог этот флажок надо установить, чтобы отобразился список содержащихся в нём файлов.

В режиме File Ages файлы отображаются в порядке убывания давности их модификации или попадания в репозиторий. При этом рядом с каждым файлом отображается примечание к точке сохранения, в которой он был обнаружен (рис. 9.30).

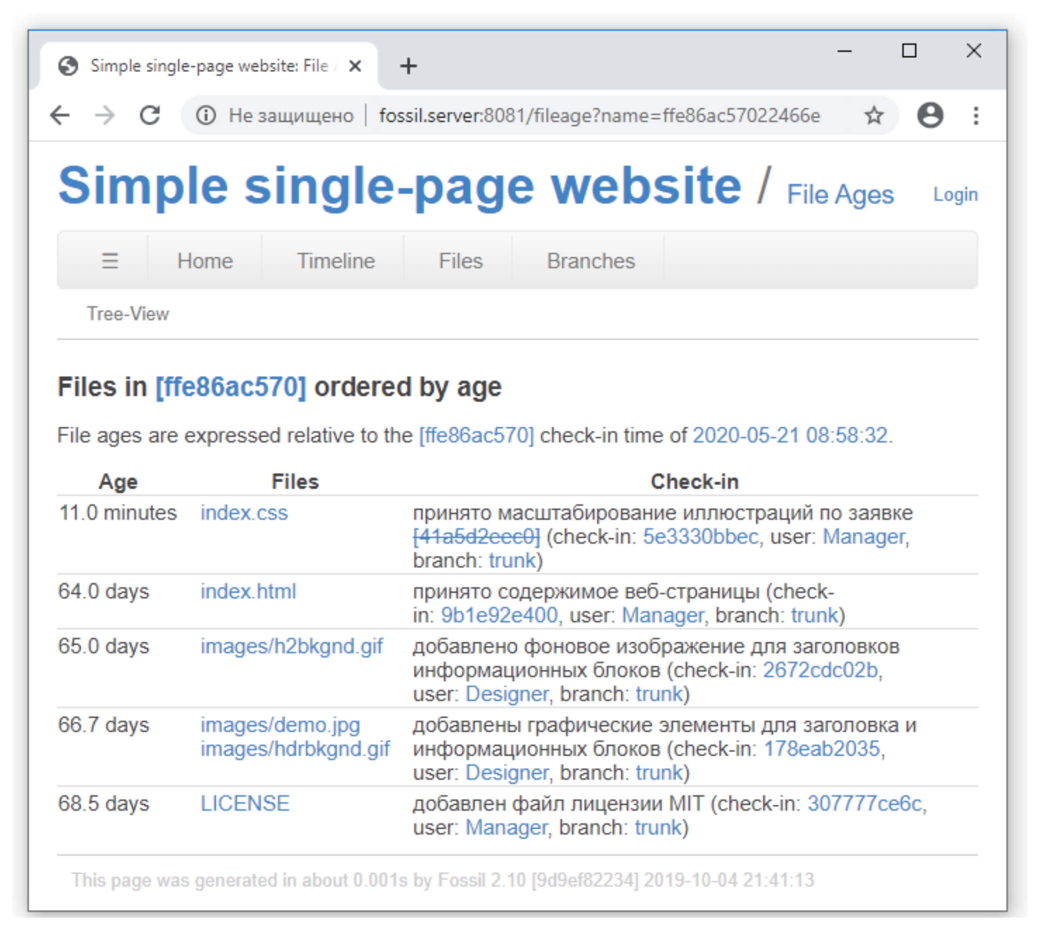


Рис. 9.30

Отображение файлов в виде таблицы в порядке убывания времени модификации, устанавливаемое пунктом меню File Ages

9.8.2. Операции над файлами

Во всех рассмотренных режимах просмотра имена файлов являются гиперссылками, с помощью которых можно перейти в режим их просмотра. Система Fossil умеет определять типы файлов по их расширению и по умолчанию отображает их наиболее естественным образом. Например, файл index.html отображается как веб-страница, получившаяся в соответствии с содержащимся в файле HTML-кодом. Поскольку при этом не используются подключаемые стили, то результат выглядит не совсем презентабельно. В рассматриваемом примере лучше воспользоваться пунктом дополнительного меню Text, чтобы перейти в режим просмотра исходного текста (рис. 9.31). В этом режиме можно установить флажок Line Numbers, чтобы отображались номера строк. Вернуться в предыдущий режим можно с помощью пункта меню Html. Пункт меню Hex даёт возможность включить режим просмотра шестнадцатеричных кодов, который может оказаться полезным для двоичных файлов.

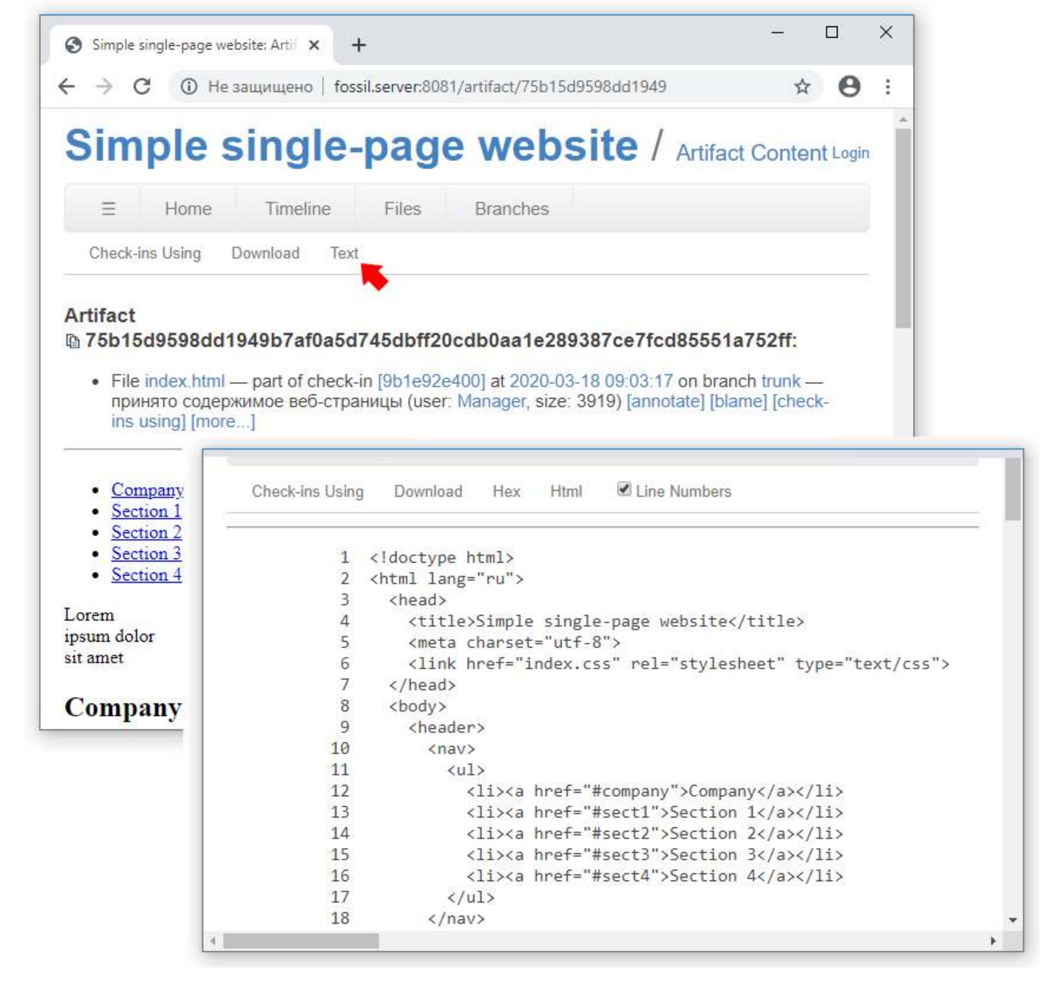


Рис. 9.31

В режим просмотра исходного текста файла можно переключиться с помощью пункта меню Text*, а для отображения номеров строк — установить флажок* Line Numbers

С помощью пункта меню Check-ins Using можно переключиться в режим отображения шкалы времени, на которой будут показаны только те точки сохранения, в которых присутствует выбранный файл.

Наконец, пункт меню Download даёт возможность выгрузить выбранный файл из репозитория. Однако работа этого пункта меню имеет одну особенность. Поскольку при его выборе приводится в действие гиперссылка на выбранный файл, то веб-браузер, настроенный на отображение файлов определённых типов, вместо предложения сохранить его на диск может просто показать содержимое файла в своём окне.

Если требуется именно сохранить файл в файловую систему компьютера, то надо на пункте меню Download с помощью правой кнопки мыши вызвать контекстное меню веб-браузера и выбрать в нём пункт Сохранить ссылку как... (рис. 9.32).

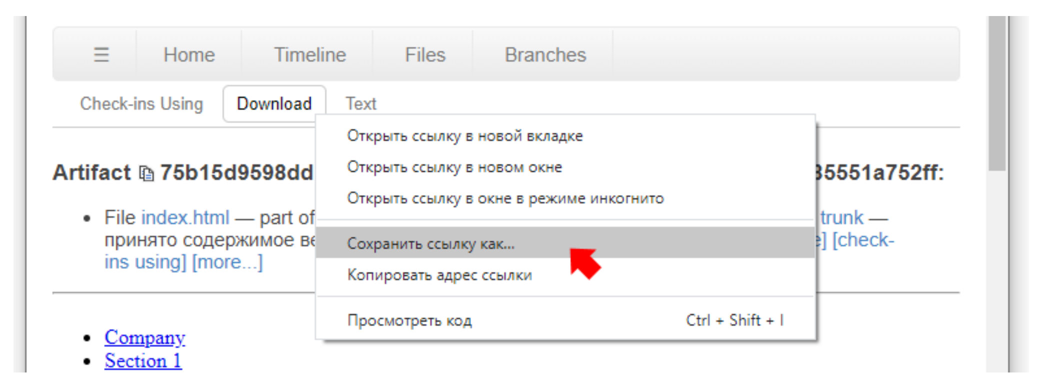


Рис. 9.32

Для сохранения файла надо вызвать контекстное меню на пункте Download *и выбрать в нём* Сохранить ссылку как...

В целом можно сказать, что режим просмотра файлов предлагает альтернативный способ навигации по истории проекта, в котором ориентиром являются не точки сохранения, а файлы рабочего каталога. В некоторых ситуациях этот способ удобнее, чем использование шкалы времени.

9.9. Бесконтрольные файлы

Все рассмотренные подсистемы Fossil используют для хранения своих данных тот же механизм, который лежит в основе контроля версий. Это означает, что хранится полная история изменений каждого объекта, будь то заявка на доработку, страница документации или сообщение на форуме. И в большинстве случаев применение этого механизма оправдано как с точки зрения сбора максимально полной информации о проекте, так и с точки зрения эффективности хранения данных (дописываются только изменения, которые были внесены в тот или иной текст, а не полное его содержимое).

Однако бывают ситуации, когда историю изменений хранить нежелательно. Например, сервер Fossil удобно использовать для распространения архивов с дистрибутивами разрабатываемого продукта. Однако хранение всех вариантов двоичных файлов было бы в данном случае совершенно лишним. Всё необходимое для создания любой версии такого дистрибутива и так имеется в репозитории. А если в базе данных Fossil будут накапливаться все двоичные файлы, соответствующие каждому выпуску продукта, то её размер может недопустимо увеличиться. Другая ситуация — распространение через сервер Fossil инструментов, которые используются в работе над проектом: текстовый редактор, компилятор, отладчик и т. п.

Система Fossil даёт возможность записывать в свою базу данных файлы, минуя механизм контроля версий только для того, чтобы распространять их через свой сервер. Такие файлы называются unversioned (неверсионированными), но для благозвучности по-русски будем называть их бесконтрольными, подразумевая, что на них не распространяется контроль версий.

Предположим, что менеджер решил использовать бесконтрольные файлы для размещения на сервере дистрибутивов системы Fossil. Тогда новый член команды при подготовке своего рабочего места сможет загрузить этот инструмент прямо с веб-сайта проекта. Для этого менеджер создал отдельный каталог tools (за пределами рабочего каталога проекта) и записал в него файлы fossil-w32-2.10.zip, fossil-w64-2.10.zip и index.html. Первые два файла — это те самые дистрибутивы, а третий файл содержит HTML-код страницы загрузки.

Листинг файла index.html

```
<!doctype html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Страница загрузки инструментов</title>
</head>
<body>
<h1>Инструменты для работы над проектом</h1>
<hr>
<h2>Система контроля версий</h2>
<ul>
<li><a href="fossil-w32-2.10.zip">Fossil (Windows 32)</a></li>
<li><a href="fossil-w64-2.10.zip">Fossil (Windows 64)</a></li>
</ul>
<hr>
<a href="../">Назад к репозиторию.</a>
</body>
</html>
```

Затем менеджер сделал каталог tools текущим и выполнил команду для добавления бесконтрольных файлов в базу данных репозитория:

```
fossil uv add *.* -R ..\sspwebsite-local.fossil
```

В данной команде *.* — это маска добавляемых файлов, а с помощью параметра *-R* указывается путь к файлу репозитория Fossil. Хранилище бесконтрольных файлов в Fossil «плоское», в нём нельзя создать иерархию каталогов. Просмотреть имеющиеся в базе данных файлы можно с помощью команды:

```
fossil uv list -R ..\sspwebsite-local.fossil
```

В рассматриваемом примере она вывела такую таблицу:

```
5f403d037abe 2020-05-23 15:51:07 2258184 2258184 fossil-w32-2.10.zip
d1dbc014bf9c 2020-05-23 15:51:07 2667722 2667722 fossil-w64-2.10.zip
4e9c0b90e5d9 2020-05-23 15:51:07 520 336 index.html
```

На текущий момент бесконтрольные файлы записаны в локальный файл репозитория на рабочем месте менеджера, а для того, чтобы организовать их распространение через сервер, их надо на сервер отправить. Но бесконтрольные файлы не участвуют в синхронизации, которая происходит между локальными и сетевыми репозиториями при обычной эксплуатации системы контроля версий. Это разумно, потому что ценности такие файлы не представляют, и не имеет смысла их реплицировать на все репозитории, тем более что они могут занимать значительный объём.

Чтобы загруженные в локальный репозиторий файлы отправились на сервер, надо ввести команду:

```
fossil uv sync -R ..\sspwebsite-local.fossil
```

Однако выполнить эту операцию сможет не всякий пользователь, а только имеющий разрешение для записи бесконтрольных файлов на сервер Write Unversioned (код y). У менеджера такого разрешения не было, и его попытка завершилась неудачей:

```
Warning: uv-pull-only
Unable to push unversioned content because you lack
sufficient permission on the server
Round-trips: 1 Artifacts sent: 0 received: 0
done, sent: 352 received: 324 ip: 192.168.56.101
```

Предоставить необходимое разрешение может только владелец репозитория (прав администратора для этого недостаточно). Поэтому придётся авторизоваться в системе под именем и установить требуемую отметку в карточке пользователя Manager (рис. 9.33).

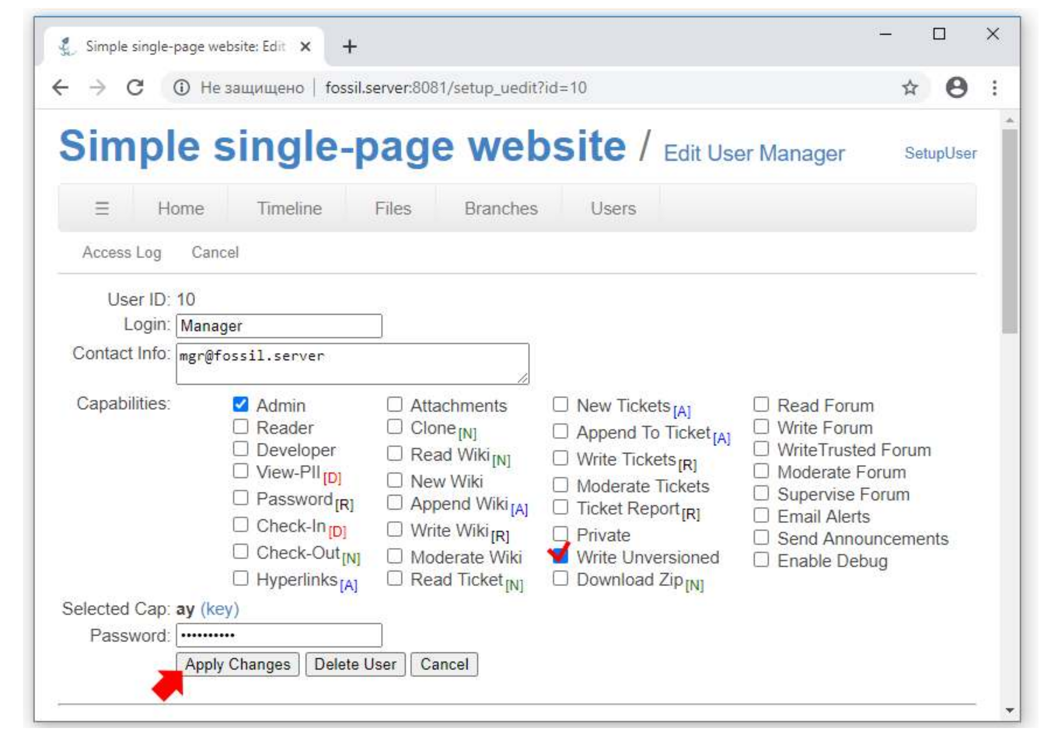


Рис. 9.33

Предоставление разрешения для отправки бесконтрольных файлов на сервер

При повторном выполнении команды информационный обмен прошёл без ошибок:

```
Round-trips: 4 Artifacts sent: 3 received: 0
done, sent: 4924731 received: 1255 ip: 192.168.56.101
```

Чтобы посмотреть, какие бесконтрольные файлы имеются на сервере, можно воспользоваться в веб-интерфейсе пунктом раскрывающегося меню Administration Pages и перейти в открывшемся окне по гиперссылке Unversioned Files (рис. 9.34).

Simple single-page website: Про ×
Не защищено | fossil.server:8081/setup

Simple single-page website / Простой одностраничный веб-сайт
Manager

☰ Home Timeline Files Branches Users

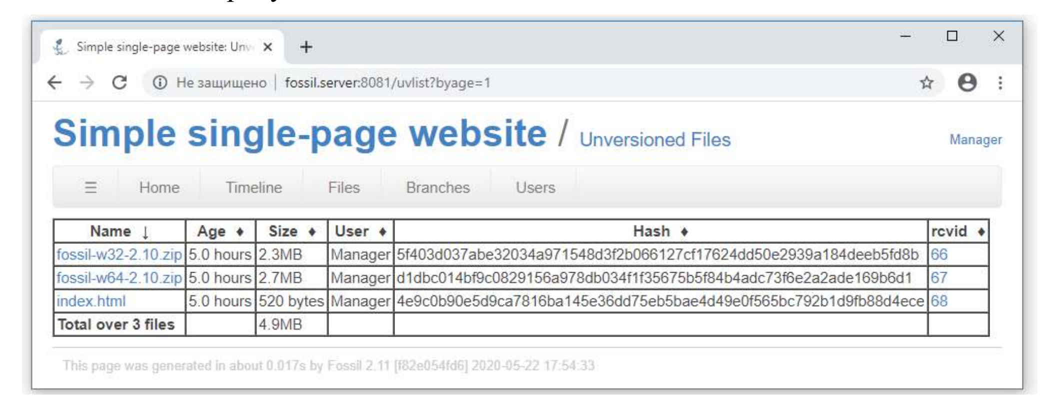
- Home Page
- File Browser
  - Tree-view, Trunk Check-in
  - Flat-view
  - File ages for Trunk
  - Unversioned Files
- Project Timeline
  - Activity Reports
  - File name changes
  - Forks
  - First 10 check-ins
- Logout
  - Change Password
- Repository Status
  - Collisions on hash prefixes
  - List of URLs used to access this repository
  - List of Artifacts
  - List of "Timewarp" Check-ins
- Administration Pages
  - Pending Moderation Requests
  - Admin log
  - Status of the web-page cache

Artifact Receipts Log	A record of received artifacts and their sources
User Log	A record of login attempts
Administrative Log	View the admin_log entries
Error Log	View the Fossil server error log
Unversioned Files	Show all unversioned files held
Stats	Repository Status Reports
Sitemap	Links to miscellaneous pages

Рис. 9.34

Переход к просмотру бесконтрольных файлов

В рассматриваемом примере список бесконтрольных файлов выглядит, как показано на рисунке 9.35.



Name ↓	Age ♦	Size ♦	User ♦	Hash ♦	rcvid ♦
fossil-w32-2.10.zip	5.0 hours	2.3MB	Manager	5f403d037abe32034a971548d3f2b066127cf17624dd50e2939a184deeb5fd8b	66
fossil-w64-2.10.zip	5.0 hours	2.7MB	Manager	d1dbc014bf9c0829156a978db034f1f35675b5f84b4adc73f6e2a2ade169b6d1	67
index.html	5.0 hours	520 bytes	Manager	4e9c0b90e5d9ca7816ba145e36dd75eb5bae4d49e0f565bc792b1d9fb88d4ece	68
Total over 3 files		4.9MB			

Рис. 9.35

Таблица со списком имеющихся на сервере бесконтрольных файлов

Если в дальнейшем потребуется обновить какой-нибудь бесконтрольный файл, надо для нового файла повторить процедуру добавления его в локальный репозиторий с помощью команды fossil add и отправки его на сервер с помощью команды fossil uv sync. При этом, как уже отмечалось, старый вариант файла в хранилище будет перезаписан и утрачен безвозвратно.

Файлы можно не только загружать в хранилище, но и удалять из него. Например, чтобы удалить из хранилища бесконтрольный файл fossil-w32-2.10.zip, надо выполнить команду:

```
fossil uv rm fossil-w32-2.10.zip -R ..\sspwebsite-local.fossil
```

Как и при добавлении, удаление произойдёт из файла локального репозитория ..\sspwebsite-local.fossil. Для удаления этого файла из хранилища сервера надо выполнить синхронизацию хранилищ:

```
fossil uv sync -R ..\sspwebsite-local.fossil
```

Базовый адрес хранилища бесконтрольных файлов на сервере имеет вид: http://fossil.server:8081/uv/. Например, если набрать в адресной строке браузера: http://fossil.server:8081/uv/index.html, то будет запрошен загруженный в это хранилище файл index.html, представляющий собой HTML-код веб-страницы загрузки инструментов (рис. 9.36). Таким образом можно создать в главном меню веб-интерфейса репозитория пункт Download, аналогичный таковому на официальном сайте проекта Fossil.

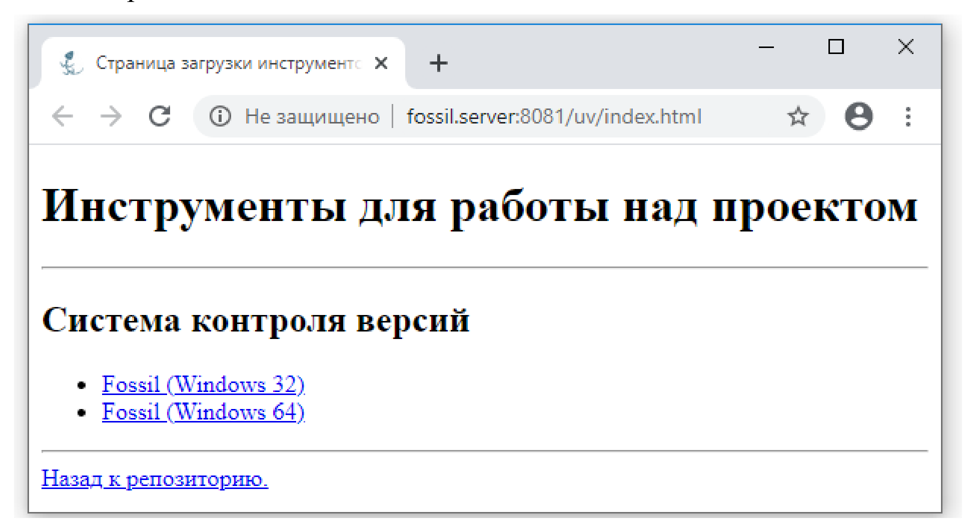


Рис. 9.36

Страница загрузки инструментов

Как уже было отмечено, бесконтрольные файлы не участвуют в обычном клонировании и синхронизации репозиториев (и это разумно). Однако участники проекта всё же имеют возможность их получить в свои локальные репозитории, если укажут параметр *-u* в командах fossil clone или fossil sync, либо выполнят специальную команду:

```
fossil uv sync -R sspwebsite-local.fossil
```

При желании в любой момент можно очистить хранилище бесконтрольных файлов в конкретном файле репозитория Fossil (например, в sspwebsite-local.fossil) с помощью команды:

```
fossil sql "DROP TABLE unversioned; VACUUM;" -R sspwebsite-
local.fossil
```

Однако такое удаление, в отличие от команды fossil uv rm, не формирует инструкции для системы Fossil, требующей распространить это действие на синхронизируемые репозитории. По этой причине, если теперь выполнить команду синхронизации хранилищ fossil uv sync, вместо очистки хранилища сетевого репозитория будет выполнено получение бесконтрольных файлов из сетевого репозитория в локальный.

ЗАКЛЮЧЕНИЕ

3.1. Организация резервного копирования

В начале этой книги описывалась вымышленная ситуация, в которой её персонаж одномоментно безвозвратно утратил все свои наработки, хотя следовал разумной рекомендации создавать и поддерживать в актуальном состоянии резервные копии важных данных. Видимо, книга должна была подсказать решение этой проблемы. Однако, если хорошо подумать, проблема по-прежнему актуальна.

Система контроля версий действительно хранит в репозитории историю изменений всех файлов проекта. В сетевом варианте использования эта информация распределяется между всеми участниками проекта. Казалось бы, всё предусмотрено. Однако корень проблемы заключается в том, что репозитории хранятся в файлах, те, в свою очередь, — на материальных носителях информации, а физические объекты подвержены поломкам.

Если в результате редкого природного явления или действия вредоносной программы в файле репозитория поменяются несколько байтов, это, возможно, никак не отразится на осуществляемой с ним работе. В репозиторий по-прежнему будут записываться новые снимки рабочих каталогов, по-прежнему будут идти обсуждения на форуме, файл репозитория по-прежнему будет регулярно копироваться на резервный носитель информации...

Но только это уже будет повреждённый файл, содержащий в себе мину замедленного действия. Возможно, она никогда не проявится. Может быть, повреждённой окажется картинка, которую легко восстановить. Но если повреждения затронут структуру базы данных SQLite, последствия трудно предсказать.

Поэтому даже при использовании системы контроля версий остаётся риск утраты ценной информации. Но система Fossil позволяет свести этот риск к минимуму, если перед выполнением резервного копирования файла базы данных репозитория проверять его целостность. Хранение данных в репозитории организовано по принципу, напоминающему популярную в мире криптовалют технологию блокчейн. Поступающий в репозиторий элемент, будь то моментальный снимок рабочего каталога, сообщение на форуме или заявка на доработку, снабжается заголовком, который называется манифестом (manifest). Помимо сведений о добавленном элементе (дата и время, описание, имя пользователя) в этот заголовок записываются контрольные суммы как добавленного элемента, так и его предков.

В результате получается цепочка, в которой контрольная сумма следующего звена зависит от содержимого предыдущего. Именно поэтому однажды записанная в репозиторий информация не может быть из него удалена — для фактического удаления элемента потребовалось бы пересчитывать контрольные суммы всех элементов, добавленных позже, т. е. перестраивать репозиторий. Но

зато такой способ хранения данных позволяет проверить их целостность и обнаружить несанкционированные изменения.

Пересчитывать контрольные суммы всех элементов и сверять их с записанными в заголовках при выполнении каждой операции с репозиторием было бы слишком накладно с вычислительной точки зрения. Поэтому для проверки целостности файла репозитория имеется специальная команда fossil test-integrity. В качестве параметра ей надо указать имя проверяемого fossil-файла. Например, для проверки базы данных репозитория проекта, рассматриваемого в этой книге, на сервере надо ввести команду:

```
fossil test-integrity sspwebsite.fossil
```

Будет пересчитана и проверена вся цепочка контрольных сумм, и в случае отсутствия ошибок программа сообщит:

```
120 non-phantom blobs (out of 120 total) checked: 0 errors
120 блоков данных (из 120 всего) проверены: 0 ошибок
low-level database integrity-check: ok
Низкоуровневая проверка целостности базы данных: ok
```

Если же такое сообщение не будет получено или будет получено другое сообщение, то это говорит о наличии повреждений в проверяемом файле. Например, подобный вывод команды должен насторожить:

```
wrong hash on artifact 117
skip phantom 164
0ee27070866cb2f9120ffd89c96eee26f09351337997e642b40988c1d030aeb9
165/167
skip phantom 165
80ea8b565ca310f524c4d2014ec7f5626cf431e4ae324c6bea574c03a3cbab9b
166/167
skip phantom 166
92a4f3de513c724b06292bf45e7b8a0773b4af641e0668851c79279460a995ae
167/167
skip phantom 167
d05d2b8cfd427f3ecaaa6f399987e24e6184f30906a616fb13f2df21a70357e4
163 non-phantom blobs (out of 167 total) checked: 1 errors
low-level database integrity-check: ok
```

В первой строке говорится о том, что контрольная сумма элемента репозитория номер 117 не прошла проверку. Но сообщение в последней строке о том, что целостность структуры базы данных не нарушена, по крайней мере оставляет надежду на восстановление информации. Внешне такой репозиторий может продолжать нормально функционировать — принимать новые моментальные снимки, поддерживать обсуждения на форуме. Однако из-за принципа организации хранения информации в репозитории (в виде отличий от предыдущего состояния) высока вероятность того, что ошибка распространится на последующие моментальные снимки.

Другой пример сообщения об ошибке при проверке файла fossil-репозитория:

```
117/122
SQLITE_CORRUPT(11): database corruption at line 68308 of
[c20a353364]
SQL: SELECT content FROM blob WHERE rid=:rid AND size>=0
SQL: SELECT rid, uuid, size FROM blob ORDER BY rid
SQLITE_CORRUPT(11): statement aborts at 6: [SELECT content FROM
blob WHERE rid=:rid AND size>=0] database disk image is malformed
SQL: SELECT content FROM blob WHERE rid=:rid AND size>=0
SQL: SELECT rid, uuid, size FROM blob ORDER BY rid
Database error: SQL error (11,11: database disk image is malformed)
while running [SELECT content FROM blob WHERE rid=:rid AND size>=0]
```

Здесь уже говорится о повреждении структуры базы данных. Скорее всего, проблемы будут заметны и при обычной работе с репозиторием. Аналогичные ошибки будут возникать при синхронизации репозиториев или выполнении других операций.

Чтобы своевременно обнаружить повреждение файла репозитория, надо регулярно выполнять проверку его целостности с помощью приведенной выше команды. А во избежание описанной в начале книги неприятной ситуации такую проверку в обязательном порядке надо выполнять перед резервным копированием файла репозитория.

Процедура резервного копирования, как правило, автоматизирована. Поэтому оценку результата проверки целостности файла репозитория удобнее производить не по текстовому сообщению, которое выводится на стандартное устройство вывода, а по коду завершения. В соответствии с принятыми соглашениями программа fossil возвращает операционной системе нулевое значение в случае успешного завершения и значение, отличное от нуля, в случае обнаружения ошибки. Тогда для выполнения резервного копирования в среде операционной системы Windows можно использовать командный файл backuprp.bat следующего содержания.

Листинг командного файла backuprp.bat

```
@ECHO OFF
ECHO Резервное копирование файла репозитория Fossil.
ECHO %1
fossil test-integrity %1
IF ERRORLEVEL 1 GOTO ERROR1
ECHO Проверка целостности завершена успешно.
COPY %1 D:\BACKUP\
IF ERRORLEVEL 1 GOTO ERROR2
ECHO Завершено успешно.
GOTO EOF
:ERROR1
ECHO *** ОШИБКА ПРОВЕРКИ ЦЕЛОСТНОСТИ РЕПОЗИТОРИЯ ***
GOTO EOF
:ERROR2
ECHO *** ОШИБКА КОПИРОВАНИЯ ФАЙЛА ***
:EOF
```

Тогда для выполнения копирования файла репозитория C:\REPOS\sspwebsite.fossil надо выполнить команду:

```
backuprp.bat C:\REPOS\sspwebsite.fossil
```

При выполнении команды в сценарий вместо переменной %1 будет передано указанное в командной строке имя файла репозитория. Команды ECHO выводят на экран информационные сообщения. Проверка целостности файла репозитория производится командой fossil test-integrity %1.

Непосредственно следующая за ней команда IF ERRORLEVEL 1 GOTO ERROR1 вызывает переход к метке ERROR1, если код завершения, возвращённый предыдущей командой, больше или равен единице, что, как было сказано, свидетельствует об ошибке проверки. Если же код завершения не больше нуля, то выполнение команд продолжается в порядке их следования в тексте сценария. Таким образом, предыдущая резервная копия будет перезаписана новой только в том случае, если копируемый файл репозитория пройдёт проверку целостности.

Вторая команда IF ERRORLEVEL 1 GOTO ERROR2 проверяет результат выполнения самой операции копирования в каталог D:\BACKUP\, которая может завершиться ошибкой, например, из-за отсутствия свободного места на целевом носителе информации или из-за сбоя подключения сетевого носителя.

Перед копированием файла сетевого репозитория, размещённого на сервере, надо останавливать работу сервера Fossil. Это требуется для того, чтобы в копию попали все завершённые транзакции и данные из буферов операционной системы. Копирование файла репозитория, с которым в этот момент работает сервер, чревато получением неполноценной копии. В следующих версиях Fossil планируется реализовать механизм, позволяющий создавать корректные копии fossil-файлов на лету с помощью команды вида:

```
fossil sql "vacuum repository into 'sspwebsite.backup'" -R sspweb-
site.fossil
```

Но пока что это лишь планы разработчиков. А до их реализации следует соблюдать правильную процедуру резервного копирования.

3.2. Восстановление информации из репозитория

Предположим, что неприятность всё же случилась — файл сетевого репозитория оказался повреждённым, проблема не была своевременно обнаружена и распространилась на резервные копии. Что можно предпринять в этом случае?

Во-первых, надо помнить о том, что регулярно синхронизируемые локальные репозитории содержат большую часть информации, имеющейся в сетевом репозитории. Возможно, наиболее простым решением станет размещение на сервере одного из таких локальных репозиториев, после чего потребуется вручную произвести его донастройку — завести учётные записи участников проекта и записать бесконтрольные файлы.

Во-вторых, можно попытаться спасти информацию из повреждённого fossil-файла. Как неоднократно отмечалось, fossil-файлы — это файлы в форма-

те базы данных SQLite, и для их обслуживания можно использовать любые инструменты, пригодные для работы с SQLite. Кроме того, саму программу fossil можно запустить в режиме работы с базой данных репозитория на низком уровне.

Допустим, при проверке оказалось, что на 117-м элементе обнаружено повреждение структуры базы данных:

```
117/122
SQLITE_CORRUPT(11): database corruption at line 68308 of [c20a353364]
…
Database error: SQL error (11,11: database disk image is malformed)
while running [SELECT content FROM blob WHERE rid=:rid AND size>=0]
```

Перед выполнением любых дальнейших действий надо в обязательном порядке создать резервную копию этого, пусть даже повреждённого, файла репозитория. Восстановительные мероприятия в условиях неопределённости не всегда приводят к успеху, а чтобы испытать другой метод восстановления, исходная копия файла очень пригодится.

Если веб-интерфейс репозитория ещё функционирует, то можно попытаться узнать, что за элемент скрывается под таким номером. Для этого надо воспользоваться пунктом раскрывающегося меню List of Artifacts (рис. 3.1).

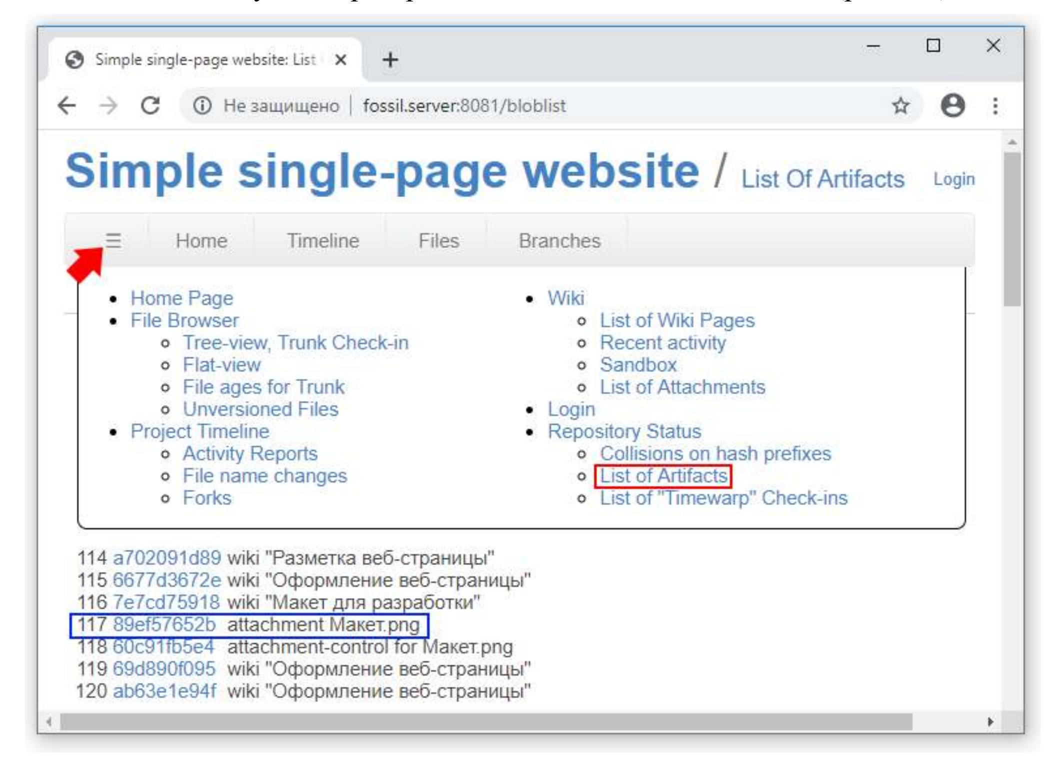


Рис. 3.1

Просмотр списка элементов репозитория

В списке элементов под номером 117 значится иллюстрация Макет.png. Но по гиперссылке с её идентификатором 89ef57652b вместо детальной информации об элементе выводится сообщение об ошибке (рис. 3.2).

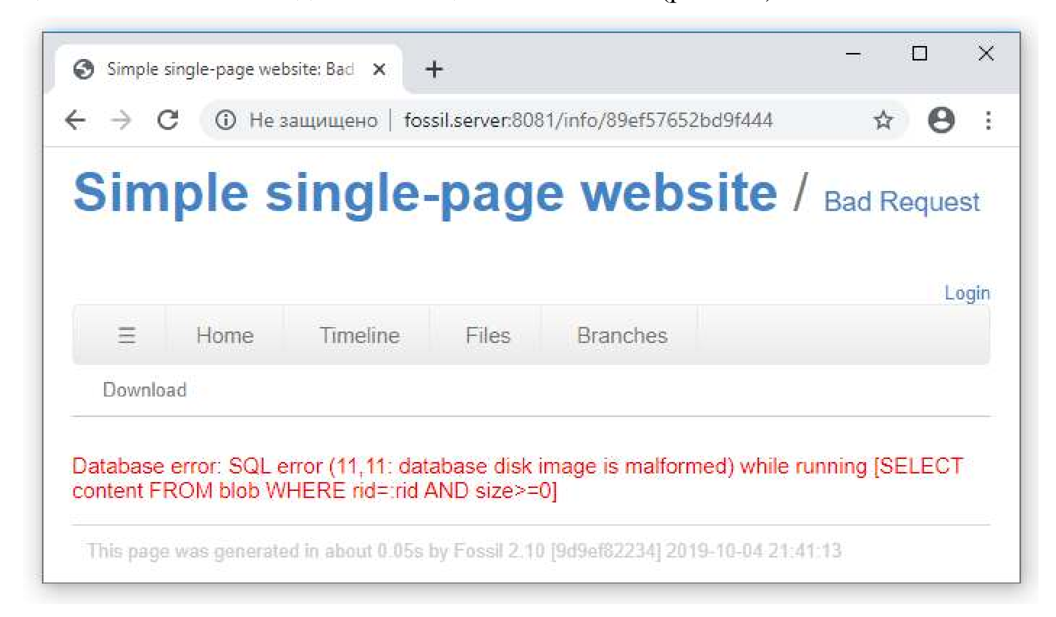


Рис. 3.2

Вместо детальной информации об элементе отображается сообщение об ошибке базы данных

Дальнейшие действия надо проводить из командной строки либо непосредственно на сервере, где размещён файл сетевого репозитория, либо переписав этот файл на локальный компьютер. Первое, что можно попробовать, — это запустить штатную процедуру перестроения базы данных репозитория с помощью команды:

```
fossil rebuild -R sspwebsite.fossil
```

В данном случае это не помогло, выполнение команды завершилось с ошибкой:

```
89.9% complete...
SQLITE_CORRUPT(11): database corruption at line 68996 of [2fc80ef16c]
…
Database error: SQL error (11,11: database disk image is malformed)
while running [SELECT content FROM blob WHERE rid=:rid AND size>=0]
```

В режиме низкоуровневой работы с базой данных можно выгрузить содержимое fossil-файла в текстовый файл, представляющий собой последовательность команд SQL, которые необходимо выполнить для создания базы данных со всем её содержимым с нуля. Для этого предназначена команда:

```
fossil sql .recover -R sspwebsite.fossil > sspwebsite.sql
```

В случае успешного выполнения на экран эта команда ничего не выводит, но рядом с fossil-файлом sspwebsite.fossil будет создан текстовый файл

sspwebsite.sql. Срествами операционной системы переименуем повреждённый файл репозитория sspwebsite.fossil в sspwebsite.fossil.bad, после чего воссоздадим структуру файла репозитория с помощью последовательности команд:

```
fossil sql -R sspwebsite.fossil.bad
sqlite> .open sspwebsite.fossil
sqlite> .read sspwebsite.sql
sqlite> .quit
```

Первая команда запускает режим низкоуровневой работы с базой данных. Поскольку этой команде необходимо указание имени файла fossil-репозитория, в параметре *-R* передаётся имя файла повреждённого репозитория, но можно, например, создать новый репозиторий и указать имя его fossil-файла. Вторая команда инициирует создание файла пустой базы данных с именем sspwebsite.fossil. Третья команда запускает выполнение SQL-команд, содержащихся в текстовом файле sspwebsite.sql, который содержит инструкции по созданию fossil-репозитория. Последняя команда предназначена для завершения работы с программой.

В результате файл репозитория sspwebsite.fossil будет создан заново. Снова выполним команду перестроения базы данных:

```
fossil rebuild -R sspwebsite.fossil
```

На этот раз она завершается без ошибок, структура базы данных восстановлена:

```
100.0% complete...
```

Но радоваться рано. Выполним команду проверки целостности репозитория:

```
fossil test-integrity sspwebsite.fossil
```

Она выводит на экран следующее сообщение:

```
wrong hash on artifact 117
Неправильная контрольная сумма элемента 117
127 non-phantom blobs (out of 127 total) checked: 1 errors
127 блоков (из 127 всего) проверено: 1 ошибка
low-level database integrity-check: ok
Низкоуровневая проверка целостности базы данных: ок
```

Структура базы данных восстановлена, но чуда не произошло и повреждение 117-го элемента не исцелилось. Если теперь запустить сервер Fossil и через веб-интерфейс перейти на страничку с информацией о 117-м элементе репозитория с идентификатором 89ef57652b, то будет выведено его описание (рис. 3.3). Однако при выборе пункта дополнительного меню Download вместо макета веб-сайта отобразится чёрный квадрат. Судя по всему, файл Макет.png в репозитории безнадёжно повреждён.

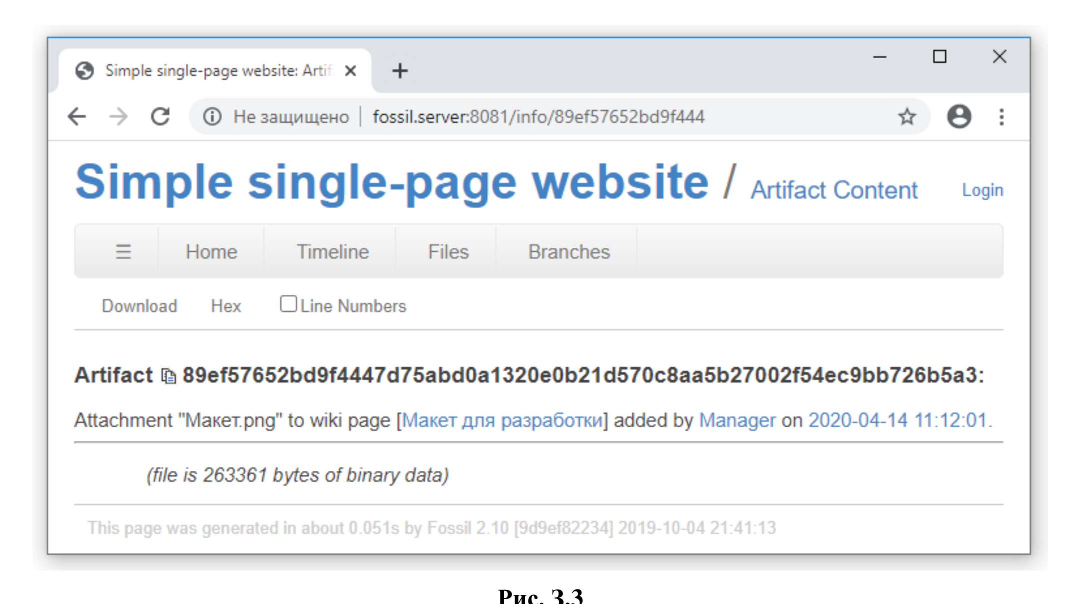


Рис. 3.3

Страница с описанием 117-го элемента репозитория

В критической ситуации существует возможность выгрузить всё содержимое репозитория в виде отдельных файлов. Сначала создадим для них каталог artifacts, после чего выполним команду:

```
fossil deconstruct --private -R sspwebsite.fossil artifacts
```

В случае успеха эта команда сообщит:

```
100.0% complete...
```

В результате работы этой команды в каталоге artifacts будут созданы подкаталоги 00, 01, 02, ..., a0, a1, ..., fe, ff, имена которых соответствуют первым двум шестнадцатеричным цифрам идентификаторов выгруженных элементов. В этих подкаталогах будут находиться файлы с именами, составленными из оставшихся цифр идентификаторов хранящихся в них элементов, начиная с третьей. Например, проблемный 117-й элемент с идентификатором 89ef57652b должен оказаться в файле с именем ef57652bd9f4..., который размещён в подкаталоге 89. И действительно, такой файл нашёлся, только он оказался нулевой длины — из-за повреждения система Fossil не смогла его выгрузить правильно.

Для выполнения обратной операции по сворачиванию выгруженных элементов репозитория из каталога artifacts в файл базы данных repo.fossil надо выполнить команду:

```
fossil reconstruct repo.fossil artifacts
```

Эта команда выводит на экран сообщение:

```
Reading files from directory "artifacts"...
127
Building the Fossil repository...
178.2% complete...
project-id: 1031fa4d21bfbcaeafef34e313f8f2a28933fe3d
```

```
server-id: 6a8f5a0b4478bd5d53a138af9db238d7265bd8a9
admin-user: PCUser (initial password is "g9wKTnzybQ")
```

В результате будет создан файл нового fossil-репозитория repo.fossil. Однако много информации из исходного репозитория, например учётные записи пользователей, будет утрачено.

Описанные примеры низкоуровневых манипуляций с fossil-репозиторием приведены в большей степени для того, чтобы показать, что восстановить повреждённый файл репозитория Fossil в первозданном виде практически невозможно. Поэтому стоит уделять внимание правильному резервному копированию сетевого репозитория, а в случае форс-мажора лучшим вариантом станет замещение сетевого репозитория одним из локальных с выполнением отсутствующих настроек.

3.3. Обновление системы

Интегрированная система Fossil — это динамично развивающийся проект. Актуальная на момент начала написания книги версия 2.10 к её завершению сменилась на 2.11. Каждая новая версия несёт в себе устранение обнаруженных ошибок, улучшение работы существующих механизмов и добавление новых функций. Поэтому у пользователей Fossil безусловно имеется интерес в поддержании своих инструментов в актуальном состоянии.

Сложность перехода на новую версию заключается в том, что система Fossil двухкомпонентная — она состоит из исполняемого файла fossil, реализующего всю логику работы, и хранилища информации — базы данных SQLite. Если обновление исполняемого файла не представляет трудности (его можно просто заменить новой версией, полученной с официального сайта), то о базе данных такого сказать нельзя. Недопустимо просто взять и переписать существующий файл репозитория — при этом пришлось бы начинать работу над проектом с самого начала.

Не удивительно, что разработчики Fossil включили механизм обновления структуры базы данных в исполняемый файл fossil. В результате правильная процедура обновления состоит из двух шагов:

1) заменить исполняемый файл fossil (fossil.exe для Windows) новой версией;

2) обновить базы данных репозиториев с помощью команды fossil all rebuild.

Эти действия надо выполнить как на сервере Fossil, так и на индивидуальных рабочих местах всех участников проекта. Команда fossil all rebuild обновляет структуры баз данных всех fossil-репозиториев, которые когда бы то ни было использовались на компьютере, где выполняется эта команда. Откуда Fossil знает и помнит об этих репозиториях?

Система Fossil незаметно ведёт реестр репозиториев, над которыми выполнялись операции, в файле %LOCALAPPDATA%_fossil, %APPDATA%_fossil или %HOMEPATH%_fossil, если работа происходит в операционной системе Windows, либо в файле .fossil домашнего каталога пользователя в случае других

операционных систем. Это даёт возможность выполнения однотипных операций сразу над всеми репозиториями.

Например, если планируется автономная работа без сетевого подключения, пользователь может синхронизировать сразу все свои репозитории с сервером с помощью команды fossil all sync. Бывает удобно также посмотреть все ещё не загруженные в репозитории изменения с помощью команды fossil all changes.

Возвращаясь к вопросу обновления версии Fossil, надо отметить, что переход на единую версию системы, безусловно, желателен, но маловероятно возникновение каких-нибудь серьёзных проблем, если это не произойдёт единовременно. Система Fossil имеет достаточно хорошо продуманную архитектуру, чтобы не допустить критического нарушения своей работы.

3.4. Индивидуальная разработка

В книге подробно рассмотрен пример использования системы Fossil в коллективной работе над проектом. Однако инструменты этой системы целесообразно применять и в индивидуальной разработке. Если индивидуальная разработка ведётся с использованием сервера Fossil, то можно последовательно следовать всем изложенным в книге рекомендациям, за тем исключением, что для работы достаточно одной учётной записи с максимальными полномочиями.

В то же время при отказе от использования сервера эксплуатация инструментов Fossil значительно упрощается с сохранением большей части функциональности. В этом случае использование сетевого репозитория исключается, а локальные репозитории можно называть просто репозиториями без риска возникновения путаницы. Рассмотрим шаги, которые необходимо выполнить в таком варианте использования.

Прежде всего систему Fossil необходимо установить на компьютере разработчика. О том, как это сделать, рассказано во второй главе этой книги. После этого можно создавать файл репозитория Fossil и рабочий каталог проекта. Предположим, что планируется работа над проектом с кодовым названием Project.

В дальнейшем изложении предполагается, что работа осуществляется в операционной системе Windows, поэтому в качестве разделителя элементов пути к файлам и каталогам используется символ обратной наклонной черты (\). В среде других операционных систем надо использовать в качестве разделителя символ наклонной черты (/).

Для хранения файлов с репозиториями лучше всего создать отдельный каталог в корне раздела файловой системы или в домашнем каталоге пользователя. Назовём этот каталог repos. Тогда потребуется выполнить следующую последовательность команд для создания каталога и файла нового репозитория project.fossil в нём:

```
mkdir repos
fossil init repos\project.fossil
```

Рабочие каталоги проектов тоже лучше размещать внутри общего каталога, предназначенного специально для этой цели. Чтобы создать общий каталог work и рабочий каталог проекта project в нём, надо выполнить команды:

```
mkdir work
mkdir work\project
```

Теперь можно установить связь между рабочим каталогом и соответствующим ему репозиторием. Для этого надо выполнить следующие команды:

```
cd work\project
fossil open ..\..\repos\project.fossil
```

На этом подготовку к работе над проектом Project можно считать завершённой. Основными элементами созданной структуры стали:

- файл репозитория repos\project.fossil;
- файл с информацией о рабочем каталоге и соответствующем ему репозитории work\project_FOSSIL_ (или work\project\.fslckout, если операционная система — не Windows).

С этого момента можно начинать работу над проектом. Чтобы система контроля версий начала отслеживание изменений в файлах с исходными текстами проекта, их надо добавить в список Fossil с помощью команды:

```
fossil add СписокФайлов
```

Запись моментального снимка рабочего каталога в репозиторий Fossil производится с помощью команды:

```
fossil commit -m "ОписаниеИзменений"
```

Откат содержимого рабочего каталога к одному из предыдущих состояний может быть выполнен с помощью команды:

```
fossil checkout ТочкаСохранения
```

Даже при индивидуальной работе над проектом не следует игнорировать возможность параллельной разработки в нескольких ветвях. Чтобы создать ответвление от одной из точек сохранения, имеющихся в репозитории, и переключить на неё рабочий каталог, надо выполнить команды:

```
fossil branch new НазваниеВетви
fossil checkout НазваниеВетви ТочкаСохранения
```

Для вливания наработок, выполненных в отдельной ветви проекта, в основную ветвь, которая называется trunk, надо выполнить команды:

```
fossil checkout trunk
fossil merge НазваниеВетви
```

Подробно о командах работы с репозиторием можно прочитать в четвёртой главе. Управлять репозиторием и изучать шкалу времени проекта удобнее всего через веб-интерфейс. Быстро получить к нему доступ можно, если в рабочем каталоге проекта выполнить команду:

```
fossil ui
```

В контексте веб-интерфейса следует упомянуть, что индивидуальному разработчику не стоит отказываться от возможностей работы с заявками и документирования проекта, которые описаны в пятой и седьмой главах соответственно. Систему работы с заявками на доработку при индивидуальном использовании Fossil можно превратить в список задач, подлежащих выполнению (to do list).

Важным аспектом в индивидуальной работе над проектами является организация регулярного резервного копирования файлов репозиториев (которые в рассматриваемом примере хранятся в каталоге repos) на сменные носители информации или в сетевое файловое хранилище, ведь естественное дублирование истории изменений в локальные репозитории участников проекта через сетевой репозиторий при таком варианте использования не производится. Перед выполнением копирования надо проверять целостность структуры репозитория с помощью команды:

```
fossil test-integrity project.fossil
```

Таким образом, система Fossil, предоставляющая богатые возможности при совместной работе над проектами, может оказаться полезной в повышении качества и дисциплины индивидуальной разработки.

ПРИЛОЖЕНИЯ

П.1. Глоссарий

Локальный репозиторий — файл с рекомендованным расширением .fossil, имеющий структуру базы данных SQLite, размещённый на компьютере, где ведётся работа над проектом и находится рабочий каталог проекта. Предназначен для хранения моментальных снимков рабочего каталога проекта. Может содержать привязку к сетевому репозиторию.

Сетевой репозиторий — файл с рекомендованным расширением .fossil, имеющий структуру базы данных SQLite, размещённый на сервере компьютерной сети. Предназначен для синхронизации локальных репозиториев участников проекта.

Рабочий каталог — каталог (папка), содержащий файлы, которыми управляет система контроля версий. В рабочем каталоге находится служебный файл _FOSSIL_ (или .fslckout), имеющий структуру базы данных SQLite и содержащий привязку к локальному репозиторию.

check-in — загрузка моментального снимка файлов проекта из рабочего каталога в репозиторий, создание в репозитории точки сохранения; результат такой загрузки.

check-out — выгрузка из репозитория в рабочий каталог файлов проекта в том состоянии, в котором они находились на момент создания точки сохранения; результат такой выгрузки.

commit — операция по загрузке моментального снимка рабочего каталога в репозиторий; результат такой загрузки, точка сохранения проекта, то же, что check-in.

push — отправка изменений, зарегистрированных в локальном репозитории, в сетевой репозиторий (в режиме autosync, который включён в Fossil по умолчанию, выполняется автоматически при операциях с локальным репозиторием).

pull — получение изменений, зарегистрированных в сетевом репозитории, в локальный репозиторий (в режиме autosync, который включён в Fossil по умолчанию, выполняется автоматически при операциях с локальным репозиторием).

branch — ветвь разработки, состоящая из последовательности check-in. Основная ветвь создаётся автоматически и называется trunk, но по желанию разработчика могут быть созданы дополнительные параллельные ветви для тестирования или развития определённых версий разрабатываемого проекта.

fork — нежелательное разветвление линии разработки, произошедшее из-за неустранимого конфликта внесённых в проект изменений; ветвь разработки (branch), возникшая в результате такого события.

П.2. Основные команды Fossil

П.2.1. Команды общего назначения

fossil help Команда — вывести на экран подсказку по использованию команды Команда.

fossil all list — вывести список локальных репозиториев.

fossil all push — выполнить команду push для всех известных локальных репозиториев.

fossil all pull — выполнить команду pull для всех известных локальных репозиториев.

fossil all sync — выполнить синхронизацию всех локальных репозиториев с сетевыми.

fossil all rebuild — обновление версий всех известных локальных репозиториев (рекомендуется после обновления исполняемого файла Fossil).

П.2.2. Команды над файлом репозитория

fossil init --admin-user Владелец ФайлРепозитория — создать новый репозиторий в файле ФайлРепозитория и назначить его владельцем пользователя с именем Владелец.

fossil ui ФайлРепозитория — открыть в веб-браузере страницу веб-интерфейса для репозитория ФайлРепозитория от имени его владельца.

fossil server -P Порт ФайлРепозитория — предоставить сетевой доступ к репозиторию ФайлРепозитория и ожидать сетевых подключений на TCP-порту с номером Порт.

fossil clone URL ФайлРепозитория — создать локальную копию сетевого репозитория, доступного по адресу URL, в файле ФайлРепозитория.

fossil open ФайлРепозитория — назначить текущий каталог рабочим каталогом для репозитория, находящегося в файле ФайлРепозитория, и выгрузить в него последний моментальный снимок основной ветви разработки.

fossil remote -R ФайлРепозитория URL — установить для локального репозитория в файле ФайлРепозитория привязку к сетевому репозиторию, доступному по адресу URL (можно использовать для локального репозитория при смене пароля пользователя в сетевом репозитории).

П.2.3. Команды рабочего каталога

fossil settings crlf-glob '*' — не предупреждать о наличии в файлах рабочего каталога двухбайтовых символов перевода строки CR/LF.

fossil status — показать состояние рабочего каталога.

fossil changes — показать отличия рабочего каталога от моментального снимка.

fossil extras — показать файлы, находящиеся в рабочем каталоге, но не входящие в список файлов, управляемых системой контроля версий.

fossil diff — показать детальные различия между моментальным снимком в репозитории и текущим состоянием рабочего каталога.

fossil add ИмяФайла — внести файл ИмяФайла в список файлов, управляемых системой контроля версий.

fossil rm ИмяФайла — исключить файл ИмяФайла из списка файлов, управляемых системой контроля версий.

fossil mv --hard СписокФайлов Каталог — переместить файлы, перечисленные через пробел в списке СписокФайлов, в каталог Каталог, как в системе контроля версий, так и в файловой системе.

fossil commit -m "Описание" — Загрузить моментальный снимок файлов рабочего каталога в репозиторий с комментарием Описание (создать в репозитории точку сохранения — check-in).

fossil amend Идентификатор -m "Описание" — заменить описание точки сохранения Идентификатор на новое Описание.

fossil checkout Идентификатор — привести содержимое рабочего каталога в соответствие с точкой сохранения, хранящейся в репозитории и имеющей идентификатор Идентификатор (выгрузить из репозитория в рабочий каталог моментальный снимок check-in с идентификатором Идентификатор).

fossil update — привести файлы проекта, находящиеся в рабочем каталоге, в соответствие с содержимым репозитория с сохранением внесённых в них изменений.

fossil push — отправить изменения локального репозитория в сетевой репозиторий (в режиме автоматической синхронизации, установленном по умолчанию, выполняется автоматически).

fossil pull — получить изменения сетевого репозитория в локальный репозиторий (в режиме автоматической синхронизации, установленном по умолчанию, выполняется автоматически).

fossil sync — обменяться изменениями между локальным и сетевым репозиторием (в режиме автоматической синхронизации, установленном по умолчанию, выполняется автоматически).

fossil undo — отменить последнюю команду.

fossil redo — повторить выполнение отменённой команды.

П.2.4. Команды над ветвями

fossil branch — вывести список имеющихся в репозитории ветвей разработки.

fossil branch new "Название" Идентификатор — создать в репозитории новую ветвь разработки с названием Название, взяв за её основу моментальный снимок, соответствующий точке сохранения Идентификатор.

fossil checkout "Название" — переключить рабочий каталог на ветвь Название.

fossil merge Идентификатор — выполнить слияние — наложение моментального снимка из точки сохранения Идентификатор на текущее состояние.

fossil leaves — вывести список последних точек сохранения на активных ветвях разработки.

П.2.5. Команды над ярлыками

fossil tag list — вывести список назначенных ярлыков.

fossil tag add --propagate "Название" Идентификатор — назначить распространяющийся ярлык Название точке сохранения с идентификатором Идентификатор.

fossil tag find "Название" — показать точки сохранения, отмеченные ярлыком Название.

fossil tag cancel "Название" Идентификатор — прекратить распространение ярлыка Название в точке сохранения Идентификатор.

П.2.6. Команды работы с базой данных

fossil sql .tables -R ФайлРепозитория — вывести список таблиц базы данных репозитория.

fossil sql "SELECT * FROM repository.user;" -R ФайлРепозитория — вывести записи таблицы repository.user базы данных репозитория.

П.3. Специальные файлы Fossil

%LOCALAPPDATA%_fossil или ~/.fossil — содержит местонахождение всех FOSSIL-репозиториев пользователя. Используется командой fossil all, которая полезна при выполнении синхронизации перед переходом в автономный режим разработки (fossil all pull) и после возврата в сетевой режим (fossil all push).

FOSSIL или .fslckout — находится в рабочем каталоге проекта и содержит информацию:

- о привязке рабочего каталога к локальному репозиторию;
- о точке сохранения в репозитории, которая явилась основой для файлов рабочего каталога;
- об изменениях в составе файлов рабочего каталога, которые ещё не загружены в репозиторий.

П.4. Сообщения об ошибках Fossil

current directory is not within an open checkout — текущий каталог не является рабочим каталогом проекта; не найден файл _FOSSIL_ или .fslchkout, содержащий привязку к локальному репозиторию проекта.

Error: login failed — ошибка авторизации в системе Fossil; неправильно указано имя пользователя или его пароль.

Error: not authorized to write — пользователь не имеет права на запись в сетевой репозиторий; пользователю необходимо предоставить группу разрешений Developer (код v) или отдельное разрешение Check-In (код i).

not found: trunk — в репозитории не обнаружена основная ветвь под названием trunk; возможно, основная ветвь была переименована.

П.5. Основные команды операционных систем

Действие	Windows	UNIX и Linux
Просмотреть содержимое каталога	dir ИмяКаталога	ls ИмяКаталога
Создать каталог	mkdir ИмяКаталога	
Сменить текущий каталог	cd ИмяКаталога	
Удалить пустой каталог	rmdir ИмяКаталога	
Копировать файл	copy Откуда Куда	cp Откуда Куда
Переместить файл	move Откуда Куда	mv Откуда Куда
Удалить файл	del ИмяФайла	rm ИмяФайла

Примечания:

- каталог предыдущего уровня обозначается двумя точками ..;
- в именах файлов могут использоваться подстановочные символы «звёздочка» (* — заменяет собой любое количество фактических символов) и «вопросительный знак» (? — заменяет собой ни одного или один символ);
- в качестве разделителя имён в длинном пути к каталогу или файлу в операционной системе Windows используется символ обратной наклонной черты \, а в операционных системах Unix и Linux — символ прямой наклонной черты /;
- путь от корня файловой системы должен начинаться с символа-разделителя, о котором сказано в предыдущем пункте. В противном случае путь отсчитывается от текущего каталога.

П.6. Описание учебного примера

Состав файлов учебного примера:

- index.html — текстовый файл в кодировке UTF-8, содержащий HTML-код содержимого веб-страницы;
- index.css — текстовый файл в кодировке UTF-8, содержащий CSS-код стилевого оформления веб-страницы;
- images/hdrbkgnd.gif — изображение логотипа размером 144×144 пикселей, которое размещается в правом верхнем углу веб-страницы;
- images/demo.jpg — изображение иллюстрации информационного блока размером 320×240 пикселей, которое размещается сбоку от содержимого информационного блока;
- images/h2bkgnd.gif — изображение размером 2×8 пикселей жёлтого цвета, которое служит элементом для построения линий подчёркивания заголовков информационных блоков.

Листинг файла index.html

```
<!doctype html>
<html lang="ru">
<head>
```

```
<title>Company card</title>
<meta charset="utf-8">
<link href="index.css" rel="stylesheet" type="text/css">
</head>
<body>
<header>
<nav>
<ul>
<li><a href="#sect1">Company</a></li>
<li><a href="#sect2">Section 1</a></li>
<li><a href="#sect3">Section 2</a></li>
<li><a href="#sect4">Section 3</a></li>
</ul>
</nav>
<div>
Lorem<br>
ipsum dolor<br>
sit amet<br>
</div>
</header>
<main>
<section>
<header>
<h2><a id="sect1">Company</a></h2>
<img src="images/demo.jpg" alt="demo">
</header>
<div><!-- ЗАПОЛНИТЕЛЬ --></div>
<hr>
</section>
<section>
<header>
<h2><a id="sect2">Section 1</a></h2>
<img src="images/demo.jpg" alt="demo">
</header>
<div><!-- ЗАПОЛНИТЕЛЬ --></div>
<hr>
</section>
<section>
<header>
<h2><a id="sect3">Section 2</a></h2>
<img src="images/demo.jpg" alt="demo">
</header>
<div><!-- ЗАПОЛНИТЕЛЬ --></div>
<hr>
</section>
<section>
<header>
<h2><a id="sect4">Section 3</a></h2>
<img src="images/demo.jpg" alt="demo">
</header>
<div><!-- ЗАПОЛНИТЕЛЬ --></div>
<hr>
</section>
```

```
</main>
<footer>
<p>Company contacts</p>
<p><span>Fb</span> <span>Le</span></p>
<p>Copyright © Company, 2020.</p>
</footer>
</body>
```

Примечание. Вместо комментариев <-- ЗАПОЛНИТЕЛЬ --> следует вставить текст: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Листинг файла index.css

```
* {margin: 0; padding: 0}
body {font: normal normal normal 100% sans-serif}
header, footer {max-width: 1000px; margin: auto; padding: 4ex 2%;
text-align: center}
header {
   background: gray;
   background: linear-gradient(to top,#DDD,#FFF);
}
header div {
   height: 160px;
   margin-top: 2ex;
   padding-left: 20%;
   background-image: url("images/hdrbkgnd.gif");
   background-position: 80% center;
   background-repeat: no-repeat;
   text-align: left;
   font: normal normal bold 1.8em sans-serif
}
main {max-width: 1000px; margin: auto;  padding: 4ex 2%; back-
ground-color: #EEE}
a {color: black; text-decoration: none}
header nav {
   font-weight: bold;
   font-decoration: undercore
}
header nav ul li a:hover {text-decoration: underline}
header nav ul li {margin-left: 2em; list-style-type: none; dis-
play: inline}
header nav ul li:first-child {margin-right: 6em}
section header h2 {
   background-image: url("images/h2bkgnd.gif");
   background-position: bottom;
   background-repeat: repeat-x
}
section {margin-top: 2ex; padding: 1ex 1em; text-align: justify}
section {margin-left: 5%; margin-right: 5%}
```

```
section {min-height: 200px; background: white}
section header {background: none; margin-bottom: 1ex}
section header img {margin-bottom: 1ex; width: 44%}
section:nth-child(odd) header h2 {float: left}
section:nth-child(odd) header img {float: right; margin-left:
1.4em}
section:nth-child(even) header h2 {float: right}
section:nth-child(even) img {float: left; margin-right: 1.4em}
section hr {clear: both; border: 0px solid}
footer {height: 140px; background: linear-gradient(to bot-
tom,#DDD,#FFF)}
footer p {margin-top: 2ex; font: italic normal bold 0.8em sans-
serif}
```

ОГЛАВЛЕНИЕ